НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повне найменування інституту, факультету)
Автоматизованих систем обробки інформації і управління
(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

Дипломний проект
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему: «*Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів*»

Виконав: студент IV курсу, групи *ІП-63 Заєць Роман Леонідович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доцент,.к.т.н., Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент Ткач М.М. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
14.06.2020 01:09:06 EEST

Дата звіту:
14.06.2020 19:05:48 EEST

ID перевірки:
1004022421

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Zajec_ip63

ID файлу: 1004035498 **Кількість сторінок:** 75 **Кількість слів:** 9716 **Кількість символів:** 81034 **Розмір файлу:** 1.91 MB

11.5% Схожість

Найбільша схожість: 4.27% з джерело бібліотеки. **ID файлу:** 1003962809

6.56% Схожість з Інтернет джерелами	149	Page 77
11% Текстові збіги по Бібліотеці акаунту	450	Page 78

1.25% Цитат

Цитати	1	Page 79

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

Підміна символів

Заміна символів	4

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Зайцю Роману Леонідовичу
(прізвище, ім’я, по батькові)

1. Тема проекту «*Веб-застосування для бронювання готелів з використанням рейтингової системі оцінювання клієнтів*»
керівник проекту *Ліщук Катерина Ігорівна, доцент, к.т.н.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджена наказом по університету від “07” травня 2020 р. №1081-с
2. Термін подання студентом проекту «08» *червня 2020 року*
3. Вихідні дані до проекту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення і терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3) Аналіз якості програмного забезпечення та опис випробувань
4) Керівництво користувача

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

1) Схема структурна варіантів взаємодії

2) Схема структурна компонентів програмного забезпечення

3) Схема структурна класів програмного забезпечення

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*22.02.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*03.03.2020*	
3.	*Постановка та формалізація задачі*	*18.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*28.03.2020*	
5.	*Алгоритмізація задачі*	*04.04.2020*	
6.	*Моделювання програмного забезпечення*	*10.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*14.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*21.04.2020*	
9.	*Розробка програмного забезпечення*	*27.04.2020*	
10.	*Налагодження програми*	*12.05.2020*	
11.	*Виконання графічних документів*	*18.05.2020*	
12.	*Оформлення пояснювальної записки*	*24.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*	*02.06.2020*	
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____ Заєць Роман
(підпис)

Керівник _____ Катерина Ліщук
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6311. 045440.01.81	Пояснювальна записка	70	
3	А4	КПІ.ІП-6311. 045440.02.91	Технічне завдання	11	
4	А4	КПІ.ІП-6311. 045440.03.13	Опис програми	60	
5	А4	КПІ.ІП-6311. 045440.04.34	Програма та методика тестування	6	
6	А4	КПІ.ІП-6311. 045440.05.34	Керівництво користувача	12	
7	А3	КПІ.ІП-6311. 045440.06.99.КЕ	Креслення вигляду екранних форм	1	
8	А3	КПІ.ІП-6311. 045440.07.99.СБД	Схема бази даних	1	
9	А3	КПІ.ІП-6311. 045440.08.99.СЕ	Схема структурна варіантів використання	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6311.045440.00.90		
Розробн.		Заєць Р.Л.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Ліщук К.І.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 44 таблиці, 19 рисунків, 10 джерел.

Мета дипломного проєкту: розроблення веб-застосування для покращення процесів бронювання готелів для користувачів та зменшення певних ризиків для бізнесу.

У першому розділі було сформульовано мету проведення розробки нашого веб-застосування, актуальність, основні задачі, цілі та було проведене дослідження існуючих програмних продуктів. Описано функціональні та нефункціональні вимоги і варіанти використання.

У другому розділі було проведено моделювання, аналіз та зроблено опис архітектури програмного забезпечення. Був проведений аналіз безпеки даних ПЗ.

У третьому розділі було описано план тестування, за яким проведено аналіз якості програмного забезпечення.

У четвертому розділі було вказано вимоги для розгортання серверної та клієнтської частини ПЗ, а інструкції для користувача були наведені в додатках.

КЛЮЧОВІ СЛОВА: РЕЙТИНГОВА СИСТЕМА ОЦІНЮВАННЯ КЛІЄНТІВ, ГОТЕЛІ, ВЕБ-ЗАСТОСУВАННЯ, БРОНЮВАННЯ ГОТЕЛІВ, КОРИСТУВАЧ

ABSTRACT

Explanatory note of the diploma project consists of 4 sections, 44 tables, 19 illustrations, 10 sources.

The aim of the diploma project: development of a web application to improve the hotel booking for clients and to reduce certain risks for business.

The first section formulated the purpose of our web application, relevance, main objectives, goals and conducted a study of existing software products. Functional and non-functional requirements and uses are described.

In the second section, modeling, analysis and description of the software architecture were performed. A software data security analysis was performed.

The third section describes the test plan, according to which quality of the software was analyzed.

The fourth section noted the requirements for deploying the server and client part of the software, and user instructions were provided in the appendices

KEYWORDS: CUSTOMER EVALUATION SYSTEM, HOTELS, WEB APPLICATION, HOTELS BOOKING, USER

Пояснювальна записка
до дипломного проєкту

на тему: Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTTP – Hypertext Transfer Protocol, протокол передачі гіпертексту, призначений для забезпечення зв’язку між клієнтом та сервером. Клієнт (браузер) надсилає HTTP-запит на сервер, а сервер повертає відповідь клієнту. Відповідь містить інформацію про стан запиту, а також може містити запитуваний вміст.

N-Tire – клієнт-серверна архітектура, в якій розділяються функції представлення, основної(бізнес) логіки та зберігання даних. Найбільш поширеним різновидом багаторівневої архітектури є трирівнева .

REST – Representational State Transfer, це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, Який, як правило, використовується для побудови веб-служб. Кожна одиниця інформації визначається глобальним ідентифікатором, таким як URL. Ґрунтується на протоколі передачі даних HTTP.

Cookie – невеликий фрагмент даних, відправлений веб-сервером, який зберігається на комп'ютері користувача. Використовується браузером для аутентифікації користувача, зберігання персональних даних та налаштувань.

HMAC – Hash-based Message Authentication Code, механізм перевірки цілісності інформації, який передбачає використання спільного секретного ключа для двох клієнтів.

Unit of work – патерн проєктування, що дозволяє спростити роботу з різними репозиторіями і дає впевненість, що всі репозиторії використовуватимуть один і той же контекст даних.

Vuex – JavaScript-фреймворк з відкритим вихідним кодом для створення користувацьких інтерфейсів. Легко інтегрується в проєкти з використанням інших JavaScript-бібліотек. Може функціонувати як веб-фреймворк для розробки односторінкових додатків в реактивному стилі.

Axios – JavaScript-фреймворк, який використовується для полегшення обміну даними між клієнтом та сервером. Дозволяє легко надсилати асинхронні HTTP-запити до кінцевих точок REST та виконувати операції з CRUD.

CRUD – Create Read Update Delete, акронім, який позначає 4 основні операції над базами даних, тобто створення, зчитування, оновлення та видалення.

Kestrel – кросплатформенний веб-сервер, який за замовчуванням використовується в ASP .NET Core проєктах.

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

ВСТУП

Сьогодні складно уявити наше життя без впливу інформаційних технологій, бо вони заполонили майже усі сфери діяльності людини. Завдяки розвитку ІТ стала можливим автоматизація будь-яких процесів в нашому побуті, а саме замовлення таксі, доставка товарів, запис до лікаря та інше. Сфера готельного бізнесу не є виключенням. Кожного року з'являються певні технічні рішення, призначені покращити вже існуючі проблеми в цій галузі або створити щось унікальне.

Переважна більшість людей полюбляє мандрувати. Проте, щоб подорож була максимально комфортна й незабутня, слід подбати про усі можливі дрібниці. Бронювання номеру в готелі або хостелі є важливою частиною кожної подорожі. Саме автоматизація процесів бронювання надає можливість обрати такі апартаменти, які задовольнять усі потреби та вподобання мандрівника. Для бізнесу автоматизування замовлень дозволяє збільшити кількість заявок, прискорити їх обробку, надавати найактуальнішу інформацію користувачам та підвищити надійність бронювань.

В наш час вже існують різні веб-застосування, які здобули популярність завдяки тим чи іншим перевагам, а саме зручним інтерфейсом, системою лояльності, тощо. Проте існує така проблема, що жодне з них не містить відгуків про самого клієнта. Часом трапляється, що клієнти готелю можуть доставити певні збитки, які не є вигідними для бізнесу. Тому створення такого веб-застосування, яке б відображало рейтинг як готелів, так і клієнтів зменшило б певні ризики, пов'язані з неадекватністю або некоректною поведінкою клієнтів, для власників готелів.

Метою дипломного проєкту є розроблення веб-застосування для покращення процесів бронювання готелів для користувачів та зменшення певних ризиків для бізнесу. Зі сторони користувачів це досягається за рахунок надання першим найпопулярніших пропозицій з найактуальнішою інформацією. Зі сторони бізнесу – можливість оцінювання клієнта за певною

рейтинговою системою, підтвердження сплат та надання можливості схвалювати або скасовувати бронювання, якщо майбутній клієнт вже отримував негативні відгуки після попередніх перебувань в заброньованих апартаментах.

Завданням даного дипломного проєкту є дослідження переваг та недоліків існуючих аналогів та розробка веб-застосування, яке б спростило процеси бронювання готелів за рахунок створеної рейтингової системи оцінювання клієнтів.

Практичним значенням дипломного проєкту є програмне забезпечення, яке дозволяє здійснювати бронювання користувачам та схвалювати/скасовувати його адміністрацією готелів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Коли йде мова про здійснення будь-яких онлайн послуг, це означає, що користувач взаємодіє з інформаційною системою. Під інформаційною системою розуміється система, яка є комплексом організаційних, технічних та інших засобів для контролювання процесів, одержання, збереження, пошуку, порівняння та обробки певної інформації з метою задовольнити ті чи інші потреби користувача. Така система управління призначення для вирішення основних задач, а саме:

- розпізнавання образів;
- моделювання й перетворення даних;
- прийняття рішень, на основі певної сукупності критеріїв;
- пошук інформації. [1]

Ці задачі лежать в основі багатьох систем для автоматизації процесів, які найяскравіше представленні в інформаційних системах сфери готельного бізнесу.

Головними перевагами користування системою онлайн-бронювання є:

- економія часу, витраченого на бронювання;
- можливість електронної оплати через електронний гаманець з використанням кредитної карти;
- можливість в декілька кліків знайти усі варіанти, які потенційно задовольнять потреби користувача;
- переважна більшість веб-застосувань для здійснення бронювань мають центр підтримки клієнтів, який допоможе вирішити усі питання, які виникають у разі конфліктної ситуації.

Сьогодні дуже важко знайти таке веб-застосування, яке не мало б системи відгуків про надання певних послуг з використанням рейтингу. Тому її наявність є обов’язковою для отримання актуального фідбеку від

користувачів. На основі цього фідбеку власники можуть аналізувати як надаються послуги, чи задаволені клієнти та мають змогу корегувати слабкі місця, що є важливою частиною при веденні бізнесу.

Рейтингова система оцінювання є дуже поширеним засобом для отримання середньої оцінки якості певного товару, послуги, якості роботи людини, тощо. Популярності вона набула завдяки тому, що остаточна оцінка формується з урахуванням усіх залишених відгуків, що є найбільш об'єктивним способом отримання середньої оцінки.

Переважна більшість сервісів для бронювання надає можливість залишати коментарі на сайті с загальним враженням від перебування в тому чи іншому готелі. Також до кожного коментаря часто додається рейтингова оцінка свого враження від готелю. На основі цих коментарів й середньої оцінки, користувач робить для себе певні висновки, які можуть вплинути на прийняття рішення, щодо розгляду кандидатури готелю, як майбутнього місця перебування в подорожі.

Однак, важко знайти сервіс, який мав би рейтингову систему оцінювання для клієнтів. Тому дана дипломна робота пропонує рішення цією проблеми.

1.2 Змістовний опис і аналіз предметної області

З кожним роком можливість подорожувати стає все доступнішою для людей, тому той факт, що сфера готельного бізнесу росте та користується небаченою популярністю, не є дивним.

Вибір житла завжди було одним з найважливіших этапів у підготовці до подорожі. Числені онлайн опитування показують, що від якості готелю залежить загальне враження від здійсненної мандрівки.

В наш час переважна більшість готелів має зіркову рейтингову систему оцінювання. Від кількості зірок безпосередньо залежить якість сервісу та загальне враження.

Зіркова рейтингова система складається з п’яти рівнів якості, а саме зірок, де 1 зірка відповідає таким умовам проживання, які дозволяють переночувати з мінімальним комфортом, а 5 зірок – являє собою розкішні апартаменти з системою «all inclusive», які передбачають дуже високу якість сервісу та комфорту.

За рівнем комфорту номери в готелях поділяються на категорії:

- Suite – номери поліпшеного планування, переважно складаються з двох кімнат;
- Junjor Suite – номери покращеного планування категорії напівлюкс;
- De Luxe – номери підвищеної комфортності;
- Duplex – двоповерхові номери;
- Standart – звичайні однокімнатні номери;
- Honeymoon Room – номери для молодят.

Така варіативність у виборі типу номеру дає змогу якнайкраще задовольнити потреби користувачів.[2]

Кожного року здійснюються мільйони бронювань готелів, що було б неможливим без автоматизації цих процесів. Прямо пропорційно постійному вдосконаленню й розвитку сфери ІТ ростуть й запити до сервісів, які надають послуги для резервування. Зараз важко знайти веб-застосування для здійснення бронювань готелів, яке б не містило фільтрацію з купою критеріїв, а саме: бажані дати бронювання, кількість зірок готелю, його рейтинг, наявність сніданків/обідів/вечерь, як далеко знаходиться від центру міста, ціна, на яку кількість людей розрахований номер тощо. Велика кількість критеріїв обумовлена надати можливість користувачу знайти саме такі варіанти, які б задовольнили його потреби якнайкраще.

Розроблена в даному дипломному проєкті система рейтингового оцінювання клієнтів, сприятиме покращенню комфортного перебування в апартаментах для користувачів, та зменшенню кількості конфліктів, які іноді

виникають між клієнтами та адміністрацією готелю. Створена система є прозорою й легкою в користуванні. Після терміну перебування клієнта в номері, менеджер готелю залишає відгук, стосовно порядності цього клієнта та ставить йому рейтингову оцінку, яка формується на основі ряду критеріїв, а саме: дотримання чистоти в номері, порядність, дотримання правил готелю, тощо. Отримана рейтингова оцінка з'явиться у профілі клієнта та буде доступна до перегляду менеджерам інших готелів. Тобто, таким чином можна відсіяти тих клієнтів, які мають багато негативних відгуків, що зменшить ризики для бізнесу. Під ризиками розуміються збитки, які потенційно може понести бізнес, а саме:

- поганий настрій клієнтів, який негативно впливає на статус готелю й безпосередньо на рейтинг, що є важливим показником для будь-якого бізнесу;
- матеріальні втрати від вкрадених рушників, капців, халатів та іншого майна.

1.3 Аналіз успішних ІТ-проєктів

1.3.1 Аналіз відомих технічних рішень

Існує чимало веб-застосувань, які призначенні для полегшення та покращення процесів бронювання готелів. Вони мають дуже багато спільного, проте в кожному можна знайти певну унікальність. В умовах жорстокої та великої конкуренції, потрібно змусити користувача вибрати саме ваш продукт. Сама тому були створенні наступні технічні рішення:

- програма лояльності клієнтів. Тим користувачам, які мали найбільшу активність на сайті, або здійснювали більшу кількість бронювань, відносно інших, надаються знижки на наступні бронювання або ці користувачі отримують найбільш актуальні пропозиції, які не доступні звичайним користувачам.

- можливість власникам готелів розміщувати свої пропозиції. Завдяки такому рішенню відбулось збільшення кількості користувачів, шляхом розширення активної аудиторії.

1.3.2 Аналіз відомих програмних продуктів

Після пошуку програмних продуктів, спеціалізованих на наданні послуг бронювання готелів, були виділені два представники, а саме: Booking.com та Agoda.com.

Booking.com

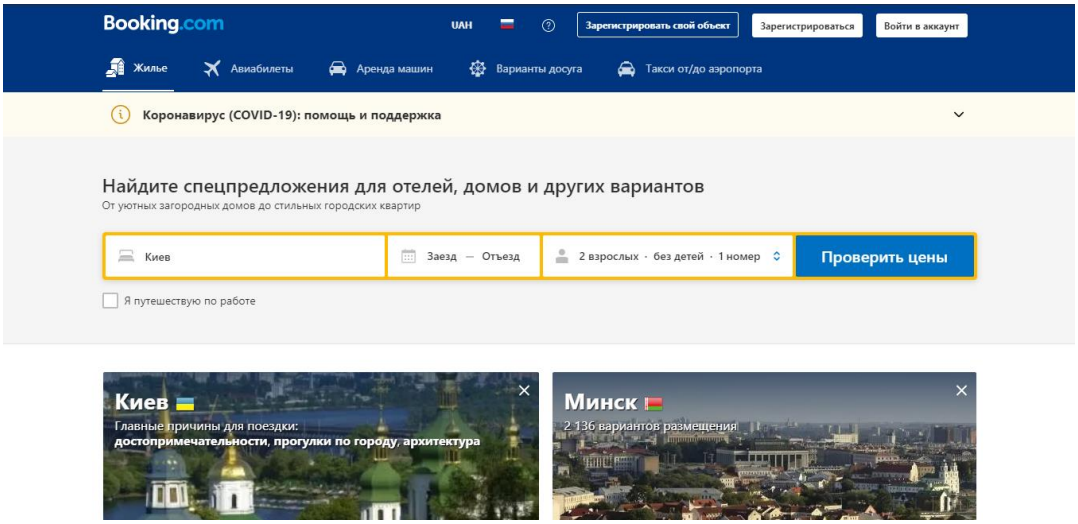


Рисунок 1.1 – Головна сторінка сайту Booking.com

Booking.com – одна з найбільших платформ для онлайн-бронювання готелів, апартаментів, туристичних подорожей, тощо. Переведена на 43 мови.

Має наступні переваги:

- дуже зручний інтерфейс;
- велика кількість різноманітних пошукових фільтрів, для задоволення будь-яких потреб користувача;
- платформа надає можливість власникам готельного бізнесу розміщувати свої пропозиції, що сприяє залученню нових користувачів;
- чудова рейтингова система оцінки якості проживання;
- зручна інформаційна дошка.

Серед недоліків можна виділити наступні:

- пропозиції підписатися на розсилку інформації;
- відсутня рейтингова система оцінювання клієнтів;
- не несе відповідальності за заброньовані готелі.[3]

Agoda.com

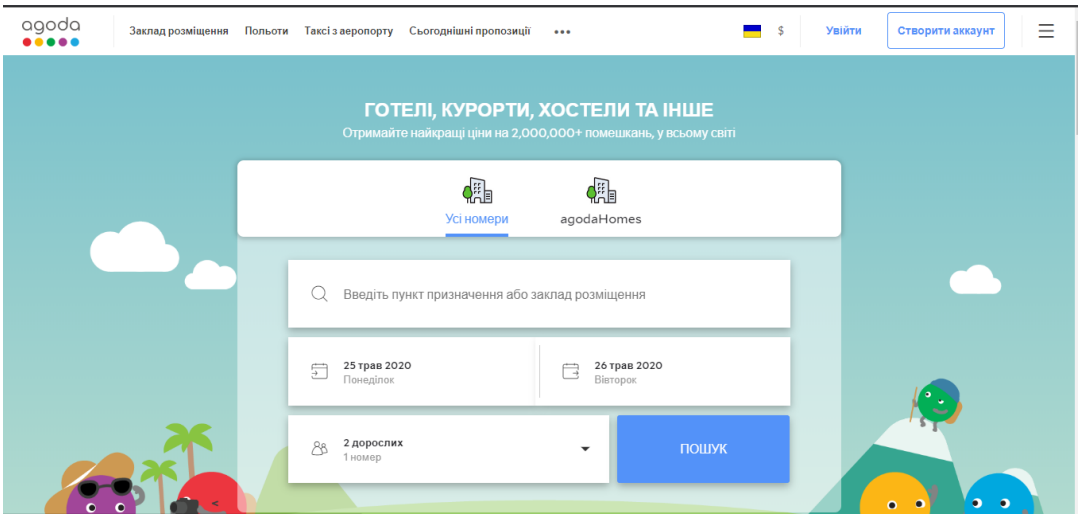


Рисунок 1.2 – Головна сторінка сайту Agoda.com

Agoda.com – одна з найбільших систем для онлайн-бронювання готелів. Переведена на 39 мов. Відмінністю даної системи бронювання є оплата повної вартості розміщення при бронюванні на більшість готелів, а також наявність програми лояльності, яка дозволяє обмінювати отримані бали на знижки і безкоштовні ночі в готелях.

Має наступні переваги:

- можливість розміщення нових пропозицій власникам готелів;
- приємний інтерфейс;
- можливість кастомізації сайту, шляхом вибору фону під свій смак.

Недоліки:

- іноді інформація про готелі не відповідає дійсності;
- відсутня рейтингова система оцінювання клієнтів;
- не несе відповідальності за заброньовані готелі.[4]

1.4 Аналіз вимог до програмного забезпечення

Аналіз вимог до програмного забезпечення необхідно розпочати з визначення ролей в веб-застосуванні та які можливості їм доступні. В ході створення моделі програмного забезпечення були виділені наступні види користувачів:

- неавторизований користувач;
- авторизований користувач;
- адміністратор;
- менеджер готелю.

Новий користувач не має змоги переглядати вміст сайту, поки не пройде етап реєстрації.

Після процесів аутентифікації та авторизації, гість стає користувачем сайту. Дана роль передбачає перегляд вмісту сайту, створення бронювання та оплату.

Адміністратор відповідає за ведення користувачів та готелів, вносить певні зміни та додає важливу інформацію.

Менеджер готелю має ті самі права, що і користувач, однак ще має можливість переглядати рейтинг користувача, скасовувати або схвалювати бронювання та оновлює статус проведення оплати за бронювання.

Розроблене веб-застосування повинне мати наступний функціонал:

- реєстрація користувача:
- авторизація користувача
- заповнення профілю користувача;
- перегляд доступних готелів;
- ведення готелів;
- фільтрації готелів за критеріями;
- ведення пошукового фільтру;
- ведення номерів готелів;
- створення бронювання;

- скасування/ухвалення бронювання;
- підтвердження оплати;
- виставлення рейтингу користувачу;

1.4.1 Розроблення функціональних вимог

Аналізуючи вимоги до ПЗ було виявлено та розділено вимоги на функціональні та нефункціональні. Функціональні вимоги – це вимоги до ПЗ, які описують поведінку системи та її внутрішню роботу. В нашому програмному забезпеченні були виявлено наступні функціональні вимоги описані в таблицях 1.1 – 1.17.

Таблиця 1.1 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Реєстрація користувача
Опис	Неавторизований користувач має змогу здійснити реєстрацію на сайті

Таблиця 1.2 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Авторизація користувача
Опис	Введений логін і пароль неавторизованого користувача повинен зберігатися в базі даних в зашифрованому вигляді, аутентифікуватися та авторизувати клієнта у разі успішності попередніх кроків

Таблиця 1.3 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Перегляд профілю користувача
Опис	Перегляд профілю доступний лише авторизованим користувачам

Таблиця 1.4 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Заповнення профілю
Опис	Заповнення профілю доступне лише авторизованим користувачам. Профіль має поля з наступною інформацією: номер мобільного, прізвище, ім’я, дата народження, інтереси.

Таблиця 1.5 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Перегляд готелів
Опис	Веб-застосування відображає список доступних готелів з детальною інформацією для авторизованого користувача.

Таблиця 1.6 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Налаштування пошукового фільтру
Опис	Форма з фільтрами відображає авторизованому користувачу усі доступні критерії для пошуку готелю. Після налаштування фільтру, система відображає ті готелі, які задовольняють умови

Таблиця 1.7 – Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Заповнення форми для бронювання
Опис	Після того, як авторизований користувач натиснув кнопку «Забронювати» на формі з вибраним готелем, система відкриває нове вікно з послідовними кроками для здійснення бронювання.

Таблиця 1.8 – Опис функціональної вимоги REQ008

Номер	REQ008
Назва	Перегляд статусу бронювання
Опис	Система відображає статус бронювання в залежності від ступеня виконання кроків для бронювання.

Таблиця 1.9 – Опис функціональної вимоги REQ009

Номер	REQ009
Назва	Додавання готелів
Опис	Адміністратор може додавати нові готелі. Додавання можливе, якщо користувач авторизований як адміністратор

Таблиця 1.10 – Опис функціональної вимоги REQ010

Номер	REQ010
Назва	Видалення готелів
Опис	Адміністратор може видаляти готелі. Видалення можливе, якщо користувач авторизований як адміністратор

Таблиця 1.11 – Опис функціональної вимоги REQ011

Номер	REQ011
Назва	Перегляд номерів готелю
Опис	Адміністратор має можливість переглядати усі номери готелю. Перегляд можливий, якщо користувач авторизований як адміністратор

Таблиця 1.12 – Опис функціональної вимоги REQ012

Номер	REQ012
Назва	Додавання номеру готелю

Продовження таблиці 1.12

Опис	Адміністратор має можливість додавати номер готелю. Додавання є можливим, якщо користувач авторизований як адміністратор

Таблиця 1.13 – Опис функціональної вимоги REQ013

Номер	REQ013
Назва	Видалення номера готелю
Опис	Адміністратор має можливість видаляти номер готелю. Видалення є можливим, якщо користувач авторизований як адміністратор

Таблиця 1.14 – Опис функціональної вимоги REQ014

Номер	REQ014
Назва	Додавання критерію до фільтру
Опис	Адміністратор має можливість додавати критерій до пошукового фільтру. Додавання є можливим, якщо користувач авторизований як адміністратор

Таблиця 1.15 – Опис функціональної вимоги REQ015

Номер	REQ015
Назва	Видалення пошукового фільтру
Опис	Адміністратор має можливість видаляти критерій з пошукового фільтру. Видалення є можливим, якщо користувач авторизований як адміністратор

Таблиця 1.16 – Опис функціональної вимоги REQ016

Номер	REQ016
Назва	Перегляд рейтингу користувача

Продовження таблиці 1.16

Опис	Менеджер готелю має змогу переглядати рейтинг користувача. Користувач повинен буду авторизований як менеджер готелю

Таблиця 1.17– Опис функціональної вимоги REQ017

Номер	REQ017
Назва	Виставлення рейтингу користувачу
Опис	Менеджер готелю має змогу виставляти рейтинг користувачу за рядом критеріїв. Користувач повинен буду авторизований як менеджер готелю

В системі передбачено наступні варіанти використання:

Таблиця 1.18 – Варіант використання UC001

Назва	Реєстрація користувача
Опис	Неавторизований користувач реєструється в системі
Учасники	Неавторизований користувач
Передумови	Відкрита сторінка з формою реєстрації користувача
Постумови	Користувач зареєстрований у системі
Основний сценарій	Користувач вводить данні у відповідні поля форми; Користувач натискає кнопку реєстрації; В разі успішної валідації відкривається головна сторінка сайту, інакше повторне введення даних.
Розширення сценаріїв	

Таблиця 1.19 – Варіант використання UC002

Назва	Авторизація користувача
Опис	Неавторизований користувач авторизується в системі

Продовження таблиці 1.19

Учасники	Неавторизований користувач
Передумови	Неавторизований користувач зареєстрований в системі. Неавторизований користувач знаходиться на сторінці авторизації
Постумови	Авторизація користувача в системі
Основний сценарій	Користувач вводить коректні данні, щоб пройти валідацію; Користувач перевіряє введені данні та натискає кнопку авторизації
Розширення сценаріїв	

Таблиця 1.20 – Варіант використання UC003

Назва	Авторизація адміністратора
Опис	Неавторизований користувач авторизується в системі як адміністратор
Учасники	Неавторизований користувач
Передумови	Неавторизований користувач зареєстрований в системі. Неавторизований користувач знаходиться на сторінці авторизації
Постумови	Користувач авторизований в системі як адміністратор
Основний сценарій	Користувач вводить коректні данні, щоб пройти валідацію; Користувач перевіряє введені данні та натискає кнопку авторизації
Розширення сценаріїв	

Таблиця 1.21 – Варіант використання UC004

Назва	Авторизація менеджеру готелю
Опис	Неавторизований користувач авторизується в системі як менеджер готелю
Учасники	Неавторизований користувач
Передумови	Неавторизований користувач зареєстрований в системі. Неавторизований користувач знаходиться на сторінці авторизації
Постумови	Користувач авторизований в системі як менеджер готелю
Основний сценарій	Користувач вводить коректні данні, щоб пройти валідацію; Користувач перевіряє введені данні та натискає кнопку авторизації
Розширення сценаріїв	

Таблиця 1.22 – Варіант використання UC005

Назва	Заповнення профілю користувача
Опис	Користувач заповнює свій профіль
Учасники	Авторизований користувач
Передумови	Авторизований користувач перейшов в свій профіль
Постумови	Користувач заповнив свій профіль інформацією
Основний сценарій	Користувач, знаходячись на головні сторінці, натискає кнопку «Профіль», переходить на нову веб-сторінку, заповнює усі поля валідними даними, натискає кнопку «Зберегти зміни»
Розширення сценаріїв	

Таблиця 1.23 – Варіант використання UC006

Назва	Перегляд доступних готелів
Опис	Користувач переглядає доступні для бронювання готелі
Учасники	Авторизований користувач
Передумови	Авторизований користувач знаходиться на головній сторінці
Постумови	Користувач знаходиться на сторінці з готелями
Основний сценарій	Авторизований користувач, знаходячись на головній сторінці, натискає кнопку «Готелі», після чого переміщується на нову сторінку з готелями для подальшого перегляду
Розширення сценаріїв	

Таблиця 1.24 – Варіант використання UC007

Назва	Ведення готелів
Опис	Адміністратор додає/видаляє готель
Учасники	Адміністратор
Передумови	Користувач авторизувався як адміністратор та знаходиться на головній сторінці
Постумови	Адміністратор додав новий готель
Основний сценарій	Неавторизований користувач авторизувався як адміністратор, перейшов на головну сторінку, натиснув кнопку «Готелі», перейшов на сторінку з готелями, натиснув кнопку «Додати готель», заповнив поля валідною інформацією, натиснув кнопку «Зберегти»

Продовження таблиці 1.24

Розширення сценаріїв	

Таблиця 1.25 – Варіант використання UC008

Назва	Фільтрація готелів за критеріями
Опис	Авторизований користувач здійснює фільтрацію готелів за певними критеріями
Учасники	Авторизований користувач
Передумови	Авторизований користувач знаходиться на сторінці готелів
Постумови	Користувач відфільтрував готелі за обраними критеріями
Основний сценарій	Користувач натискає на кнопку фільтрації, з випадаючого списку обирає необхідні критерії, натискає кнопку «Пошук»
Розширення сценаріїв	

Таблиця 1.26 – Варіант використання UC009

Назва	Ведення пошукового фільтру
Опис	Адміністратор додає/видаляє критерій з пошукового фільтру
Учасники	Адміністратор
Передумови	Користувач авторизувався як адміністратор знаходиться на сторінці з готелями
Постумови	Адміністратор додав/видалив критерій з пошукового фільтру
Основний сценарій	Адміністратор натиснув кнопку фільтрації, на випадаючому списку натиснув кнопку «Змінити», для

Продовження таблиці 1.26

	додавання натискає кнопку для додавання критерію та вводить назву в поле/для видалення натискає кнопку видалення критерію, натискає кнопку «Зберегти»
Розширення сценаріїв	

Таблиця 1.27 – Варіант використання UC010

Назва	Ведення номерів
Опис	Адміністратор додає/видаляє номери готелів
Учасники	Адміністратор
Передумови	Користувач авторизувався як адміністратор знаходиться на сторінці з номерами
Постумови	Адміністратор додав/видалив номер готелю
Основний сценарій	Для додавання адміністратор натиснув кнопку «Додати» , вводить дані кімнати в поле та натискає кнопку «Зберегти». Для видалення адміністратор натиснув кнопку «Видалити», вводить ID кімнати та натискає кнопку «Зберегти».
Розширення сценаріїв	

Таблиця 1.28 – Варіант використання UC011

Назва	Створення бронювання
Опис	Авторизований користувач створює форму для бронюванням готелю
Учасники	Авторизований користувач

Продовження таблиці 1.28

Передумови	Користувач знаходиться на сторінці з готелями
Постумови	Користувач створив форму для бронювання та очікує підтвердження від менеджеру готелю
Основний сценарій	Після фільтрації готелів, користувач обирає той, яких йому найбільше сподобався та натискає кнопку «Забронювати», переходить на сторінку з формою для бронювання, вводить необхідні данні та натискає кнопку «Відправити», статус бронювання з «Нове» змінюється на «Розглядається»
Розширення сценаріїв	

Таблиця 1.29 – Варіант використання UC012

Назва	Схвалення/скасування бронювання
Опис	Менеджер готелю схвалює або скасовує бронювання користувача
Учасники	Менеджер готелю
Передумови	Користувач авторизувався як менеджер готелю
Постумови	Менеджер готелю схвалив/скасував бронювання
Основний сценарій	Неавторизований користувач авторизується як менеджер готелю, переходить на головну сторінку, натискає кнопку «Активні бронювання», переходить на сторінку з активними бронюваннями, обирає одну з активних бронювань, передивляється оцінку користувача та відгуки, в залежності від прийнятого рішення натискає кнопку «Схвалити» у разі позитивного рішення, та кнопку «Скасувати» у разі негативного. Якщо менеджер скасував бронювання, воно зникає з сторінки с

Продовження таблиці 1.29

Основний сценарій	активними бронюваннями, а у користувача статус бронювання з «Розглядається» змінюється на «Скасовано». Якщо менеджер схвалив бронювання, статус бронювання у користувача змінюється з «Розглядається» на «Очікує оплати».
Розширення сценаріїв	

Таблиця 1.30 – Варіант використання UC013

Назва	Підтвердження оплати
Опис	Менеджер готелю підтверджую оплату бронювання
Учасники	Менеджер готелю, користувач
Передумови	Користувач оплатив бронювання
Постумови	Менеджер підтвердив оплату бронювання
Основний сценарій	Після зміни бронювання на «Очікує оплати» користувач натискає в формі з бронюванням кнопку «До оплати», відкривається форма з оплатою, користувач вводить данні кредитної картки, вносить необхідну суму та натискає кнопку «Оплатити», після цього статус бронювання змінюється з «Очікує бронювання» на «Підтвердження бронювання». Менеджер перевіряє чи прийшов платіж від користувача, в разу успіху натискає на чекбокс з назвою «Оплачено», статус бронювання змінюється на «Заброньовано», бронювання зникає з сторінки активних бронювань.
Розширення сценаріїв	

Таблиця 1.31 – Варіант використання UC014

Назва	Виставлення рейтингової оцінки користувачу
Опис	Менеджер готелю виставляє оцінку користувачу
Учасники	Менеджер готелю
Передумови	Менеджер готелю знаходиться на сторінці клієнтів
Постумови	Менеджер поставив оцінку користувачу
Основний сценарій	Менеджер, знаходячись на головні сторінці, натискає кнопку «Користувачі», переходить на сторінку з користувачами, обирає форму з користувачем та натискає кнопку «Оцінити», на форму оцінювання напроти кожного критерію в поле вводить оцінку від 0 до 10, після виставлення оцінок натискає кнопку «Підтвердити»
Розширення сценаріїв	

На рисунку 1.3 зображена матриця залежності між функціональними вимогами застосунку та варіантами використання

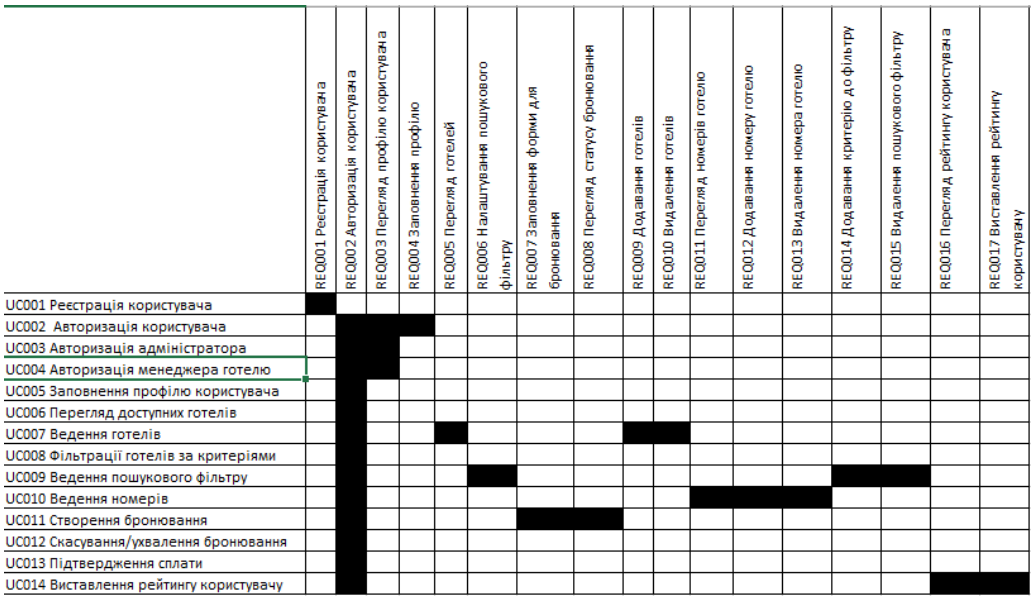


Рисунок 1.3 – Матриця залежності між функціональними вимогами застосунку та варіантами використання

1.4.2 Розроблення нефункціональних вимог

Програмне забезпечення повинне відповідати наступним нефункціональним вимогам:

— мова додатку - англійська;

— доступ до Інтернет;

— підтримка браузераів: Mozilla Firefox, Google Chrome, Safari, Internet Explorer, Opera Browser;

— операційна система Windows OS, MacOS, Linux;

— шифрування даних здійснюється за допомогою алгоритму HMAC-SHA256;

— взаємодія між сервером та клієнтом відбувається за допомогою HTTP запитів;

— не перевантажений інтерфейс користувача. Будь-яка бажана дія зі сторони користувача повинна досягатися в межах трьох натискань кнопка миші;

— швидка та надійна робота веб-застосування. Застосунок повинен відповідати нормам по швидкості функціонування програмних продуктів та бути стійким до збоїв в роботі системи;

— інтерфейс веб-застосування повинен бути простим і зрозумілим звичайному користувачу.

1.4.3 Постановка комплексу завдань модулю

Метою дипломного проєкту є створення веб-застосування для покращення і полегшення процесів по здійсненню онлайн-бронювань готелів та для зменшення певних ризиків для бізнесу.

Головна мета створення даного програмного забезпечення це створення веб-застосування для покращення процесів бронювання для користувачів та зменшення певних ризиків для бізнесу за рахунок рейтингової системи оцінювання клієнтів. Зі сторони користувачів це

досягається за рахунок надання першим найпопулярніших пропозицій з найактуальнішою інформацією. Зі сторони бізнесу - це можливість оцінювання клієнта за певною рейтинговою системою, підтвердження сплат та можливість підтверджувати або скасовувати бронювання, якщо майбутній клієнт може спричинити певні ризики. Під ризиками розуміються збитки, які потенційно може понести бізнес, а саме: поганий настрій інших клієнтів, який негативно впливає на статус цього бізнесу, певні матеріальні втрати і, як наслідок, зниження загального рейтингу самого готелю.

Для досягнення заданої мети, було виділено ряд основних завдань:

- розробка серверу для веб-застосування;
- реєстрація та авторизація користувачів;
- розмежування доступу на сайті (створення рольової моделі доступу);
- ведення готелів, номерів готелів;
- пошук готелів та номерів за заданими користувачем параметрами;
- підтвердження сплат;
- розробка чату;
- розробка рейтингової системи оцінювання авторизованих користувачів, яка формується за рахунок відгуків, залишених менеджерами готелів, в яких перебували клієнти.

На рисунку 1.4 зображена схема структурна варіантів використання.

Рисунок 1.4 – Схема структурна варіантів використання

1.5 Висновки по розділу

В даному розділі було сформульовано мету проведення розробки нашого веб-застосування, актуальність, основні задачі та цілі. Було проведене дослідження існуючих програмних продуктів, з'ясовано їх головні переваги та недоліки. Виконано аналіз відомих технічних рішень, вимог до програмного забезпечення.

Сформовано перелік функціональних та нефункціональних вимог до програмного забезпечення.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для моделювання архітектури та бізнес-процесів була використана методологія створення діаграм BPMN. [5]

Серед основних бізнес-процесів в даній дипломній роботі можна виділити наступні:

- реєстрація користувача:
- авторизація користувача, адміністратора, менеджера готелю;
- додавання адміністратором готелю;
- видалення адміністратором готелю;
- додавання адміністратором номеру готелю;
- видалення адміністратором номеру готелю;
- фільтрація готелів користувачем.

Послідовний опис реєстрації користувача в системі :

- користувач вводить логін та пароль та відсилає їх на сервер, де останній перевіряє чи є вже зареєстрований користувач з таким логіном та в разі помилки повідомляє про помилку;
- дані додаються до бази даних, що відповідає процесу реєстрації користувача. У разі виникнення помилки при додаванні, сервер повідомляє про це;
- користувач зареєстрований та переведений на сторінку авторизації.

На рисунку 2.1 зображена схема бізнес-процесу реєстрації користувача в системі.

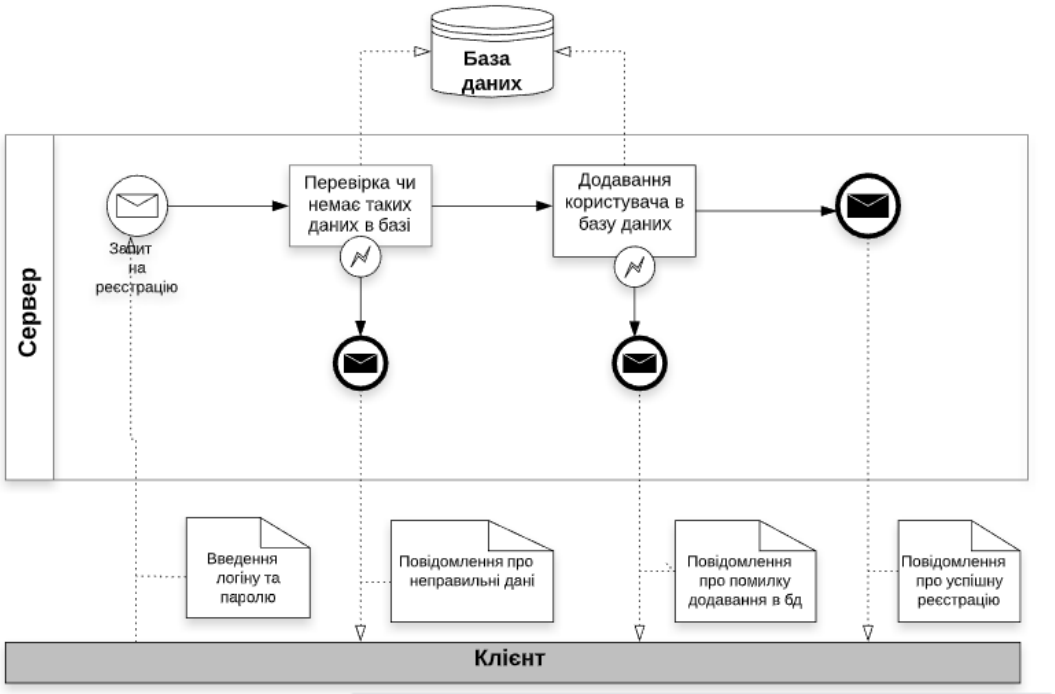


Рисунок 2.1 – Схема бізнес-процесу реєстрації користувача в системі

Послідовний опис авторизації користувача, адміністратора та менеджера готелю до сайту:

- неавторизований користувач вводить логін та пароль;
- виконується запит на сервер;
- сервер порівнює введені дані з даними в базі даних, та в разі виникнення помилки повідомляє;
- у випадку безпомилкової валідації дані користувача запам’ятовуються та виконується перенаправлення на голову сторінку сайту.

На рисунку 2.2 зображена схема бізнес-процесу авторизації користувача в системі.

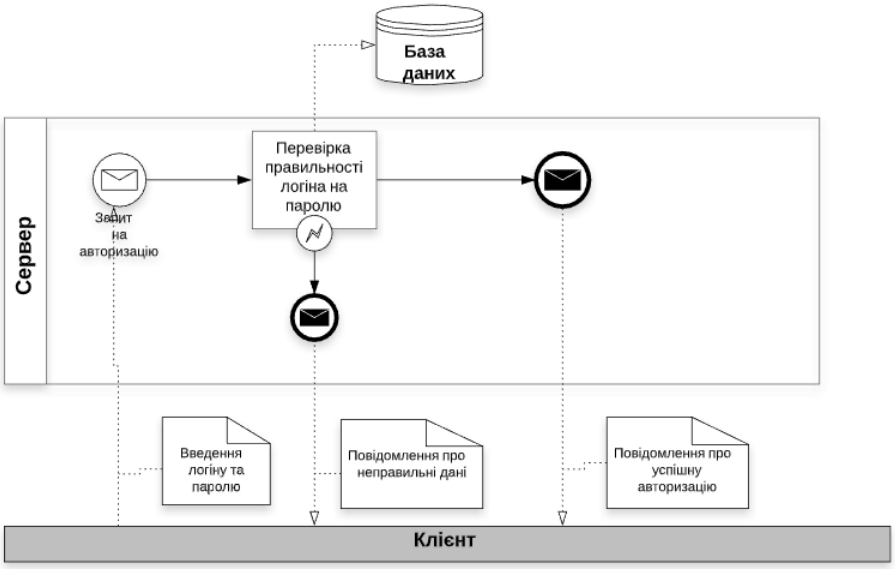


Рисунок 2.2 – Схема бізнес-процесу авторизації користувача в системі

Послідовний опис додавання готелю:

- адміністратор вносить необхідні дані готелю;
- виконується запит на сервер;
- сервер перевіряє дані і в разі помилки повідомляє;
- виконується додавання готелю в базу даних;
- після додавання готелю в базу даних, сервер повідомляє про успіх.

На рисунку 2.3 зображена схема бізнес-процесу додавання адміністратором готелю.

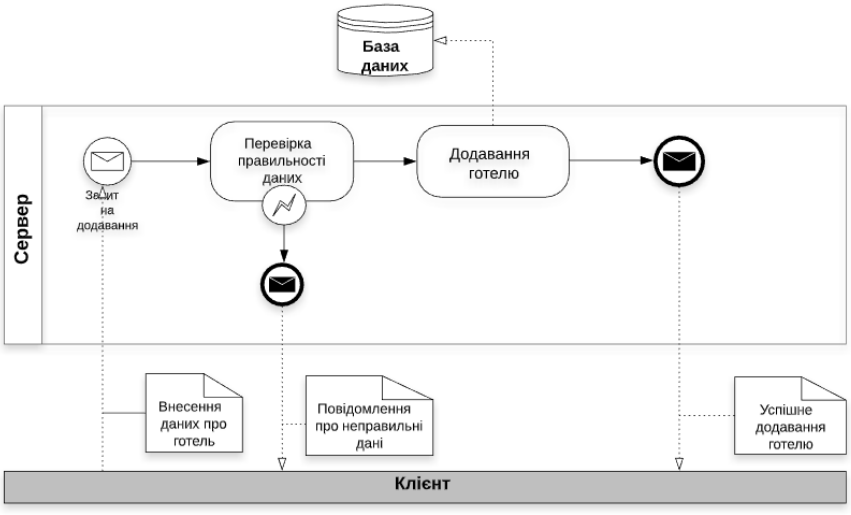


Рисунок 2.3 – Схема бізнес-процесу додавання адміністратором готелю

Послідовний опис видалення готелю;

- адміністратор вибирає готель для видалення;
- виконується запит на сервер на видалення готелю;
- сервер виконує пошук готелю в базі даних і у випадку виникнення невдачі повідомляє про помилку;
- виконується видалення готелю з бази даних;
- після видалення готелю з бази даних сервер повідомляє про успіх.

На рисунку 2.4 зображена схема бізнес-процесу видалення адміністратором готелю.

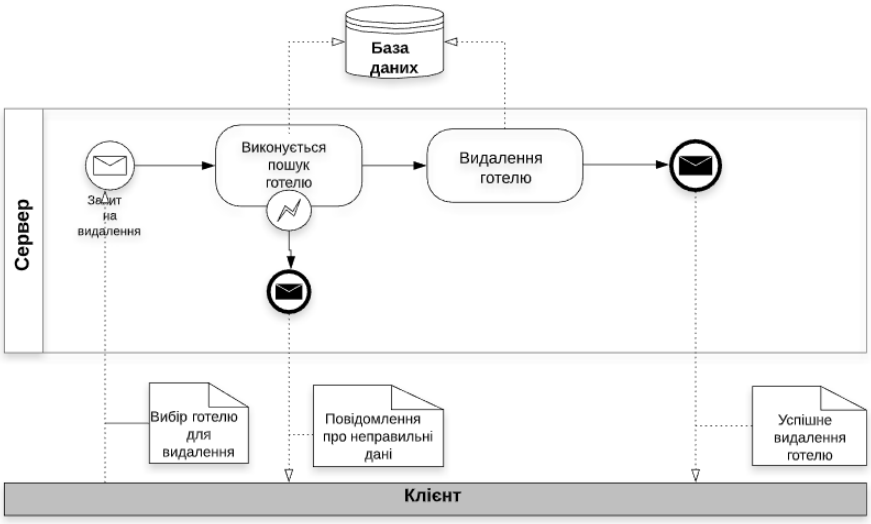


Рисунок 2.4 – Схема бізнес-процесу видалення адміністратором готелю

Послідовний опис видалення кімнати готелю;

- адміністратор вибирає кімнату для видалення;
- виконується запит на сервер на видалення кімнати;
- сервер виконує пошук кімнати в базі даних і у випадку виникнення невдачі повідомляє про помилку;
- виконується видалення кімнати з бази даних ;
- після видалення кімнати з бази даних сервер повідомляє про успіх.

На рисунку 2.5 зображена схема бізнес-процесу видалення адміністратором кімнати готелю.

Рисунок 2.5 – Схема бізнес-процесу видалення адміністратором кімнати готелю

Послідовний опис додавання кімнати готелю:

- адміністратор вносить необхідні дані про кімнату;
- виконується запит на сервер;
- сервер перевіряє дані і в разі помилки повідомляє;
- виконується додавання кімнати готелю в базу даних;
- після додавання кімнати в базу даних, сервер повідомляє про успіх.

На рисунку 2.6 зображена схема бізнес-процесу додавання адміністратором кімнати готелю.

Рисунок 2.6 – Схема бізнес-процесу додавання адміністратором кімнати готелю

Послідовний опис отримання відфільтрованого списку готелів користувачем:

- користувач знаходиться на сторінці с готелями, натискає на кнопку фільтру, вибирає бажані критерії для фільтрації та натискає кнопку «Застосувати»;
- виконується запит на сервер;
- сервер повертає список відфільтрованих готелів;
- користувач бачить список відфільтрованих готелів.

На рисунку 2.7 зображена схема бізнес-процесу отримання відфільтрованого списку готелів користувачем.

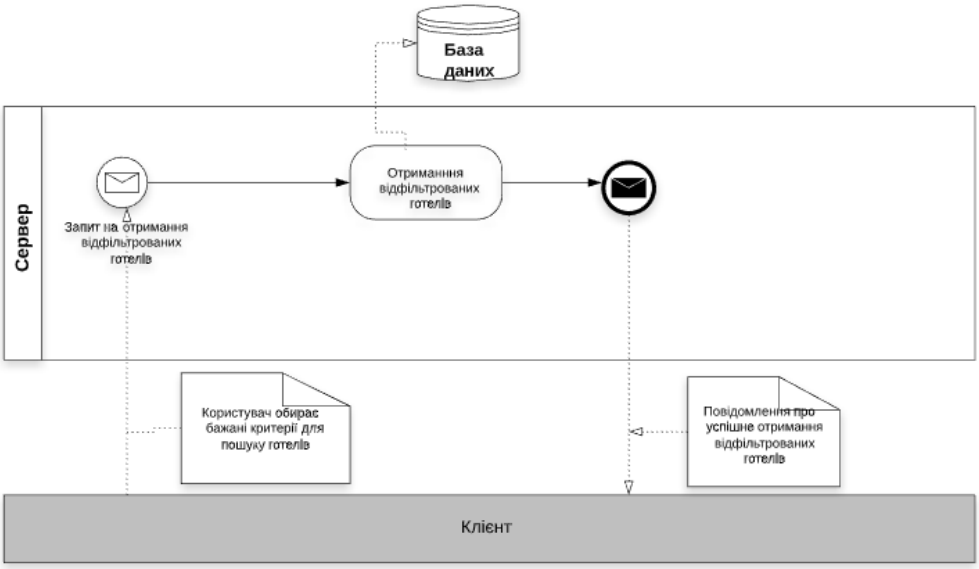


Рисунок 2.7 – Схема бізнес-процесу отримання відфільтрованого списку готелів користувачем

Послідовний опис процесу створення бронювання:

- користувач відкриває форму для створення бронювання;
- заповнює форму даними, натискає кнопку «Забронювати»;
- виконується запит на сервер;
- сервер перевіряє дані;
- виконується створення сутності бронювання в базі даних і у випадку помилки, сервер повідомляє. Статус бронювання наразі «Очікує підтвердження»;
- менеджер готелю схвалює або скасовує бронювання і натискає кнопку на формі бронювання «Схвалити» або «Скасувати»;
- виконується запит на сервер;
- сервер оновлює дані в базі даних, статус бронювання змінюється на «Очікується оплата»;
- форма бронювання оновлюється;
- користувач заповнює поля для оплати бронювання та натискає «Оплатити»;
- виконується запит на сервер;

- сервер оновлює дані в базі даних
- на оновленій формі менеджер готелю підтверджує оплату за бронювання і натискає кнопку «Підтвердити оплату»;;
- виконується запит на сервер;
- сервер виконує фінальне оновлення даних, статус бронювання змінюється на «Успішний»;
- форма бронювання оновлюється.

На рисунку 2.8 зображена схема бізнес-процесу процесу бронювання готелю користувачем.

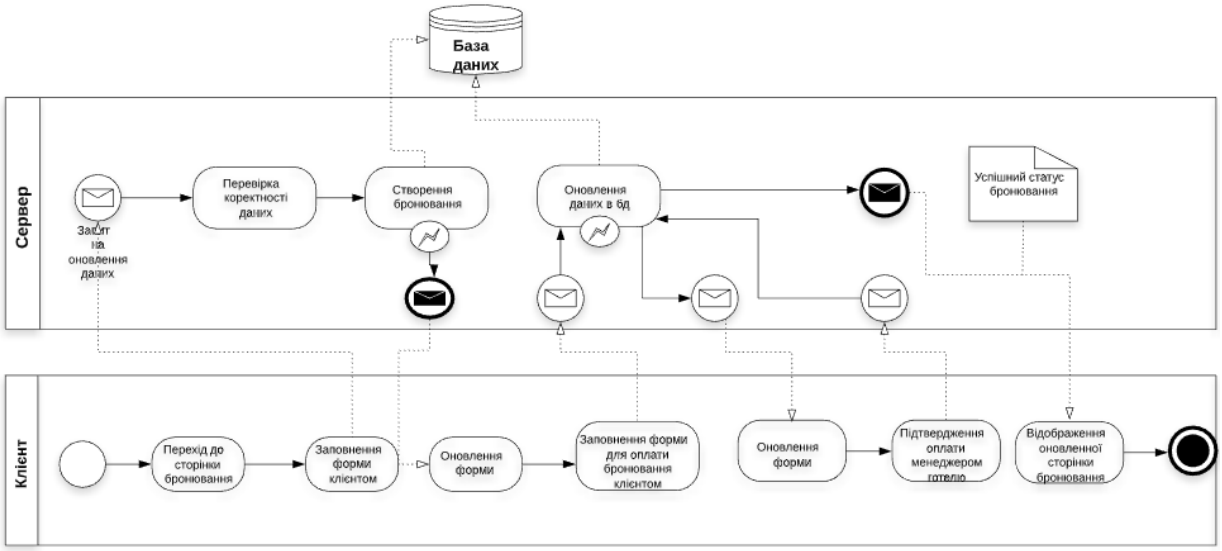


Рисунок 2.8 – Схема бізнес-процесу бронювання готелю користувачем

Послідовний опис процесу додавання рейтингової оцінки користувачу:

- менеджер готелю відкриває форму додавання рейтингової оцінки для користувача;
- заповнює форму, виставляючи оцінки певні критерії і натискає кнопку «Додати оцінку»;
- виконується запит на сервер
- сервер перевіряє введені дані та у разі виникнення помилки повідомляє;
- у разі успіху на попередньому кроці, виконується додавання рейтингової оцінки користувача в базу даних;

- після додавання сервер повідомляє про успішне виконання операції.

На рисунку 2.9 зображена схема бізнес-процесу процесу додавання рейтингової оцінки користувачу.

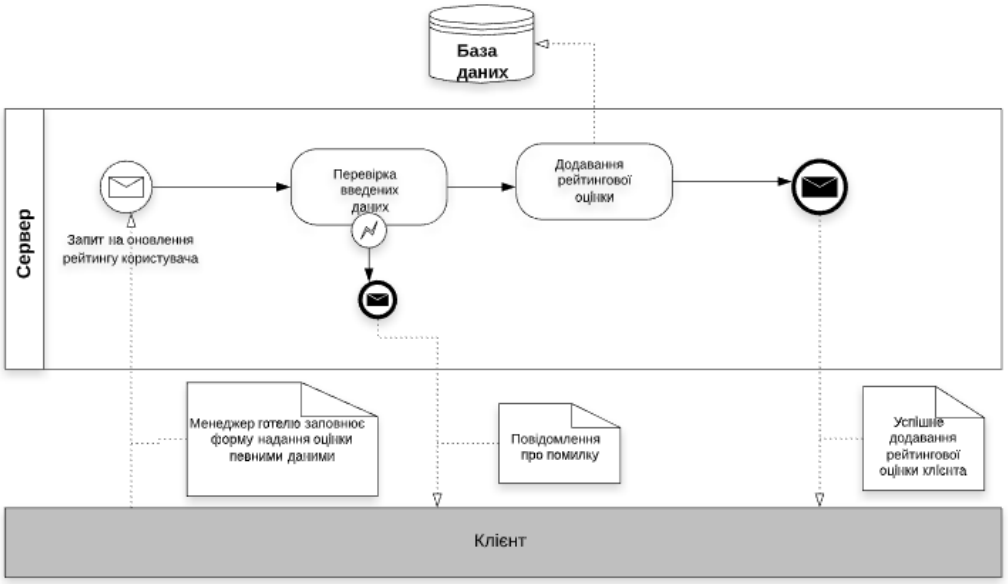


Рисунок 2.9 – Схема бізнес-процесу додавання рейтингової оцінки користувачу

2.2 Архітектура програмного забезпечення

Вибір мови програмування є одним з перших та найважливіших кроків при написанні будь-якого програмного забезпечення. Для реалізації серверної частини нашого веб-застосування було выбрано мову програмування C# на платформі .Net Core 3.1. В основі написанного серверу лежить N-Tire архітектура.

На рисунку 2.9 зображена N-Tire архітектура.

Рисунок 2.10 – N-Tire архітектура

Перший шар це DAL(Data Access Layer) – рівень доступу до даних. Другий шар це BLL(Business Logic Layer) – являє собою сукупність головної логіки нашого ПЗ. Третій шар це PL(Presentation Layer) – рівень, з яким безпосередньо взаємодіє користувач. Рівень доступу до даних не залежить від інших рівнів, рівень бізнес-логіки залежить від рівня доступу до даних, а рівень представлення залежить від рівня логіки бізнесу. Компоненти, як правило, повинні бути слабко зв'язані, тому введення DI(Dependendency Injection) є невід'ємною частиною багаторівневих додатків.[6]

В основі написання PL рівня лежить WEB API, який призначений для роботи в стилі REST (Representation State Transfer). REST-архітектура передбачає застосування таких методів або типів запитів HTTP для взаємодії з сервером:

- GET;
- POST;
- PUT;
- DELETE.

Для створення бази даних було обрано підхід Code First. Даний підхід дозволяє дуже легко та швидко створити сховище даних використовуючи лише код. В якості бази даних було вибрано Microsoft SQL Server. Для роботи з бд використовується Entity Framework Core.

Для забезпечення взаємодії репозиторіїв з єдиним контекстом даних, був використаних патерн об'єктно-реляційної логіки Одиниця роботи (Unit of Work)

Для написання клієнта було використано JS фреймворк Vue з використанням бібліотеки Vuex та Axios.

2.3 Конструювання програмного забезпечення

В основі програмного забезпечення лежить N-Tire архітектура, яка має рівень доступу до даних DAL, рівень бізнес логіки BLL та рівень взаємодії з користувачем.

При написання програмного забезпечення були виділені наступні сутності(Entity):

- готель (Hotel);
- кімната (Room);
- тип кімнати (RoomType);
- клієнт (Client);
- резервація (Reservation);
- критерії рейтингу оцінювання (RatingCriteria);
- тип харчування в готелі(NutritionType).

В таблиці 2.1 представлені репозиторії рівня DAL. Репозиторій безпосередньо взаємодії з базою даних.

Таблиця 2.1 – Репозиторії рівня DAL

Назва	Функції
ClientRepository	Додавання клієнта Видалення клієнт Оновлення клієнта в Отримання клієнта Отримання усіх клієнтів
HotelRepository	Додавання готелю Видалення готелю Оновлення готелю Отримання готелю. Отримання усіх готелів
RoomRepository	Додавання кімнати готелю Видалення кімнати готелю Оновлення кімнати готелю Отримання кімнати готелю Отримання усіх кімнат готелю
RoomTypeRepository	Додавання типу кімнати готелю Видалення типу кімнати готелю Оновлення типу кімнати готелю Отримання типу кімнати готелю Отримання усіх типів кімнат готелю.
ReservationRepository	Додавання бронювання Видалення бронювання Оновлення бронювання Отримання бронювання Отримання усіх бронювань Отримання клієнту бронювання

Продовження таблиці 2.1

	Отримання кімнати бронювання
RatingCriteriaRepository	Додавання критеріїв рейтингу Видалення критеріїв рейтингу Оновлення критеріїв рейтингу Отримання критеріїв рейтингу. Отримання критеріїв рейтингу
UnitOfWork	Виконує зв’язок репозиторіїв сутностей з базою даних. Забечпечує взаємодії репозиторіїв сутностей з єдиним контекстом даних.
NutritionTypeRepository	Додавання типу харчування Видалення типу харчування Оновлення типу харчування Отримання типу харчування Отримання типу харчування

На рисунку 2.11 зображена схема структурна класів репозиторіїв.

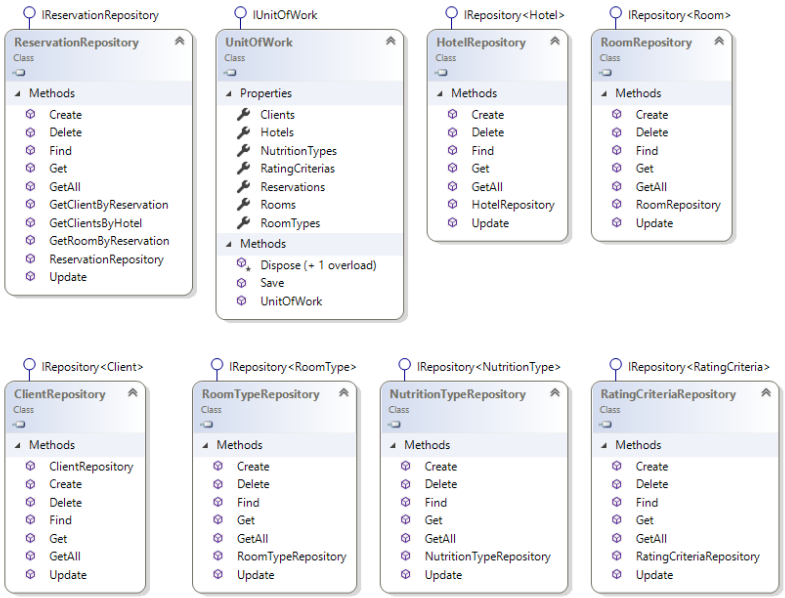


Рисунок 2.11 – Схема структурна класів репозиторіїв

Основна логіка програмного забезпечення знаходиться на рівні бізнес логіки BLL в сервісах. Сервіси через Unit Of Work взаємодіють з репозиторіями.

В таблиці 2.2 представлені сервіси рівня BLL.

Таблиця 2.2 – Сервіси рівня BLL

Назва	Функції
HotelService	Додавання готелю Видалення готелю Оновлення готелю Отримання готелю Отримання усіх готелів Отримання готелю за ім’ям Отримання готелів за містом Отримання готелів за рейтингом
RoomService	Додавання кімнати Видалення кімнати Оновлення кімнати Отримання кімнати Отримання усіх кімнат за готелем Отримання усіх кімнат за типом
ClientService	Додавання клієнта Видалення клієнта Оновлення клієнта Отримання клієнта Отримання усіх клієнтів Отримання клієнта за ім’ям Отримання клієнтів за рейтингом
RoomTypeService	Додавання типу кімнати

Продовження таблиці 2.2

	Оновлення типу кімнати Отримання типу кімнати Отримання усіх типів кімнат Отримання типу кімнати за ціною
ReservationService	Додавання бронювання Видалення бронювання Оновлення бронювання Отримання бронювання Отримання усіх бронювань Отримання усіх бронювань за певний проміжок часу Отримання клієнта за бронюванням Отримання кімнати за бронюванням Отримання усіх певного клієнтів готелю
RatingCriteriaService	Додавання рейтингу Видалення рейтингу Оновлення рейтингу Отримання рейтингу Отримання усіх критеріїв рейтингу Отримання усіх рейтингових оцінювань за клієнтом
NutritionTypeService	Додавання типу харчування Видалення типу харчування Оновлення типу харчування Отримання типу харчування Отримання усіх типів харчування

На рисунку 2.12 зображена схема структурна класів сервісів.

Рисунок 2.12 – Схема структурна класів сервісів

PL рівень використовує технологію Web API, яка надає можливість взаємодії клієнта та сервера через HTTP-запити, а саме:

- HttpGet – отримання;
- HttpPut – оновлення;
- HttpPost – додавання;
- HttpDelete – видалення.

В таблиці 2.3 представлені контроллери рівня PL.

Таблиця 2.3 – Контроллери рівня PL

Назва	Функції
HotelsController	Містить HTTP-запити для додавання, оновлення, видалення, отримання готелів.
ClientsController	Містить HTTP-запити для додавання, оновлення, видалення, отримання

Продовження таблиці 2.3

	клієнтів.
RoomsController	Містить HTTP-запити для додавання, оновлення, видалення, отримання номерів готелю.
RoomTypesController	Містить HTTP-запити для додавання, оновлення, видалення, отримання типів номерів готелю.
ReservationsController	Містить HTTP-запити для додавання, оновлення, видалення, отримання бронювань.
RatingCriteriasController	Містить HTTP-запити для додавання, оновлення, видалення, отримання рейтингу користувача.
AccountController	Відповідає за реєстрацію та авторизацію користувача

На рисунку 2.13 зображена схема структурна класів контроллерів.

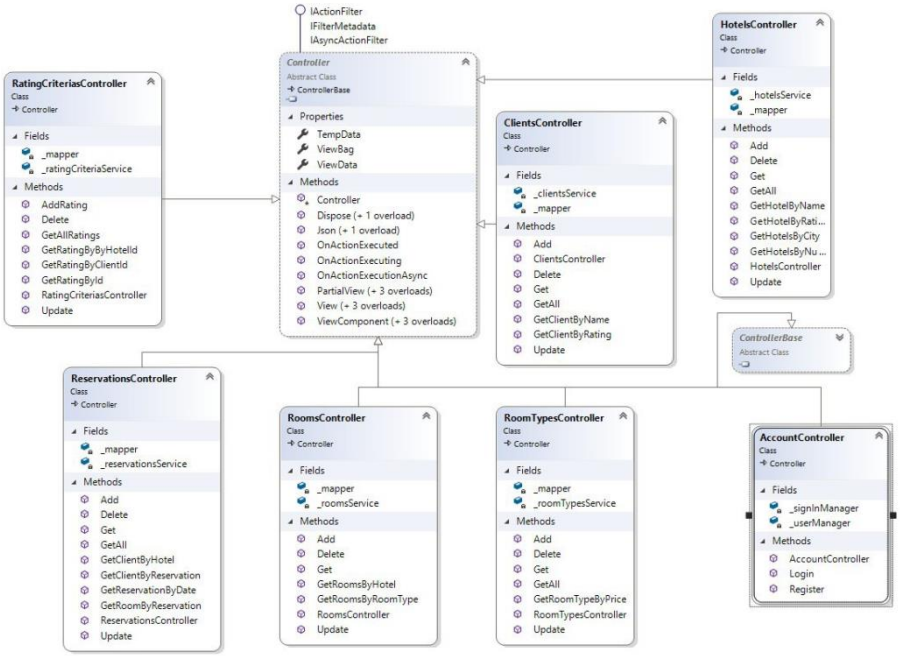


Рисунок 2.13 – Схема структурна класів контроллерів

На рисунку 2.14 зображена архітектура програмного забезпечення.

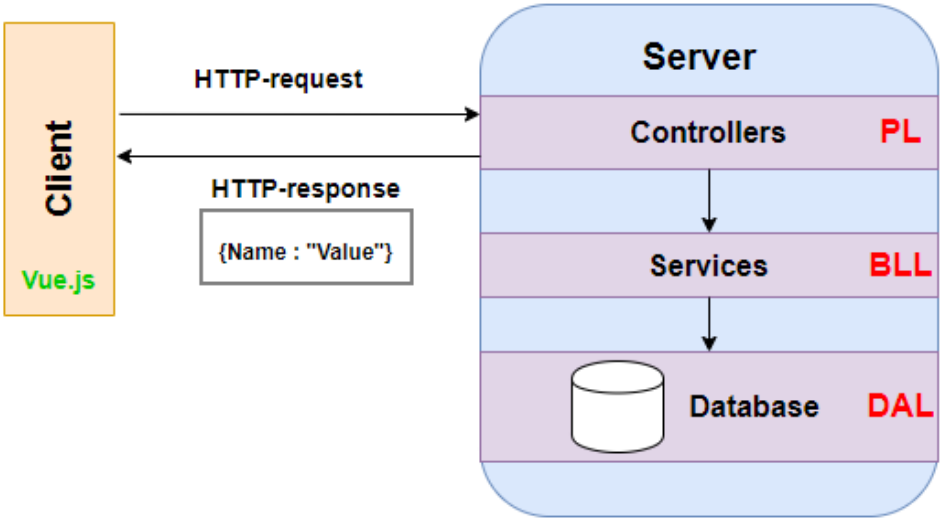


Рисунок 2.14 –Аархітектура програмного забезпечення

В таблиці 2.4 представлена специфікація методів.

Назва класу	Назва методу	Опис
AccountController	Register(RegisterModel model)	Реєструє користувача в системі, попередньо провалідувавши дані. Повертає статус операції.
	Login(LoginModel model)	Авторизує користувача в системі. Повертає статус операції.
	GetUserByEmail(string email)	Повертає інформацію про користувача за його поштовою адресою.
ClientsController	Get(int id)	Отримати клієнта за його Id
	GetAll()	Отримати усіх користувачів
	Add(ClientView client)	Додати користувача
	Update(ClientView client)	Оновити дані про клієнта

Продовження таблиці 2.4

	GetClientByName(string firstName, string lastName)	Отримати дані про користувача за його ім’ям та прізвищем
	GetClientsByRating(int rating)	Отримати усіх користувачів за рейтингом
HotelsController	Get(int id)	Отримати готель за його Id
	GetAll()	Отримати усі готелі
	Add(HotelView hotel)	Додати готель
	Update(HotelView hotel)	Оновити дані про готель
	GetHotelByName(string name)	Отримати готель за назвою
	GetHotelsByCity(string city)	Отримати готелі за містом
	GetHotelsByNutrition(int nutritionTypeId)	Отримати готелі за типом харчування
	GetHotelsByRating(int rating)	Отримати готелі за рейтингом
	GetHotelsByRoomCleaning (bool hasRoomClean)	Отримати готелі за наявність прибирання в номері
	GetHotelsByParking(bool hasParking)	Отримати готелі за наявністю місця для паркування авто
RoomsController	Get(int id)	Отримати кімнату за її Id

Продовження таблиці 2.4

	GetAll()	Отримати усі кімнати
	Add(RoomView room)	Додати кімнату
	Update(RoomView room)	Оновити кімнату
	Delete(int id)	Видалити кімнату
	GetRoomsByHotel(int hotelId)	Отримати усі кімнати вказаного готелю
	GetRoomsByRoomType(int roomTypeId)	Отримати усі кімнати за типом кімнати
RatingCriteriasController	Add(RatingCriteiasView rcv)	Додати оцінювання користувача
	GetRatingById(int ratingCriteriaId)	Отримати оцінювання користувача за Id
	GetRatingsByClientId(int clientId)	Отримати усі оцінювання заданого користувача
ReservationsController	Get(int id)	Отримати бронювання
	GetAll()	Отримати усі бронювання
	Update(ReservationView rv)	Оновити бронювання
	Delete(int id)	Видалити бронювання
	GetReservationsByDate(DateTime bookingDate, DateTime bookingDateEnd)	Отримати усі бронювання за заданий період часу
	GetClientByReservation(int reservationId)	Отримати клієнта за бронюванням
	GetClientByHotel(int hotelId)	Отримати клієнта за готелем
	GetRoomByReservation(int reservationId)	Отримати кімнату за бронювання

Продовження таблиці 2.4

Startup	ConfigureServices(IService Collection services)	Реєстрація сервісів
	Configure(IApplicationBuil era app, IWebHotesEnviron ment env)	Виконує конфігурації над отриманими запитами
DIforBLL	AddDIForBLL(this IServiceCollection services)	Реєстрація залежностей
HotelContext	OnConfiguring(DbContext OptionsBuilder builder)	Встановлення зв’язку з базою даних
	OnModelCreating(ModelBu Ilder builder)	Виконує заповнення БД даними
ModelBuilderExten sion	Seed(this ModelBuilder modelBuilder)	Заповнює БД початковими даними
UnitOfWork	Dispose(bool disposing)	Звільнення пам'яті, закриття зв'язку з БД
	Dispose()	Виклик GarbageCollector

В таблиці 2.5 представлений опис таблиць бази даних.Таблиця 2.5 – Опис таблиць бази даних

Назва таблиці	Опис
Clients	Зберігає дані про клієнтів готелю.
Hotels	Зберігає дані про готелі.
Rooms	Зберігає дані про номери готелів.
RoomTypes	Зберігає дані про типи номерів.
Reservations	Зберігає дані про бронювання.
NutritionsTypes	Зберігає дані про типи харчування в готелі.
RatingCriterias	Зберігає дані про рейтингове оцінювання клієнта.

Продовження таблиці 2.5

AspNetUsers	Зберігає дані про користувачів системи бронювання.

В таблиці 2.6 представлена структура таблиці Clients

Таблиця 2.6 – Структура таблиці Clients

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	Int	PK
FirstName	Ім’я клієнта	Nvarchar(100)	
LastName	Прізвище клієнта	Nvarchar(100)	
Rating	Рейтинг клієнта	Float	

В таблиці 2.7 представлена структура таблиці Reservations

Таблиця 2.7 – Структура таблиці Reservations

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	Int	PK
ClientId	Ідентифікатор клієнта	Int	FK
RoomId	Ідентифікатор кімнати	Int	FK
BookingDate	День початку резервування	Datetime(7)	
BookingDateEnd	День кінця резервування	Datetime(7)	

В таблиці 2.8 представлена структура таблиці Rooms

Таблиця 2.8 – Структура таблиці Rooms

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	Int	PK
RoopTypeId	Ідентифікатор типу кімнати	Int	FK
HotelId	Ідентифікатор готелю	Int	FK
IsReserved	Чи забронюована кімната(так/ні)	Bit	
Price	Вартість кімнати	Float	

В таблиці 2.9 представлена структура таблиці RatingCriterias

Таблиця 2.9 – Структура таблиці RatingCriterias

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	Int	PK
ClientId	Ідентифікатор типу клієнта	Int	FK
HotelId	Ідентифікатор готелю	Int	FK
Decency	Порядніть клієнта	Int	
RoomCleanliness	Чистота в кімнаті залишена клієнтом	Int	
HotelRulesCompliance	Дотримання правил готелю	Int	
Behavior	Загальна поведінка	Int	
Comment	Відгук	Nvarchar(100)	

В таблиці 2.10 представлена структура таблиці RoomTypes

Таблиця 2.10 – Структура таблиці RoomTypes

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	Int	PK
Name	Назва	Nvarchar(25)	
Description	Опис	Nvarchar(100)	
Capacity	Місткість	Int	

В таблиці 2.11 представлена структура таблиці Hotels

Таблиця 2.11 – Структура таблиці Hotels

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	Int	PK
Name	Назва	Nvarchar(50)	
NutritionTypeId	Ідентифікатор типу харчування	Int	FK
City	Місто	Nvarchar(100)	
Address	Адреса	Nvarchar(100)	
Description	Опис	Nvarchar(25)	
HasRoomCleaning	Наявність прибирання номеру	Bit	
HasParking	Наявність місця для паркування авто	Bit	
DistanceToCity	Відстань до центру міста	Float	
Rating	Рейтинг	Int	

В таблиці 2.12 представлена структура таблиці AspNetUsers

Таблиця 2.12 – Структура таблиці AspNetUsers

Колонка	Опис	Тип даних	Ключ
Id	Ідентифікатор	Nvarchar(450)	PK
UserName	Ім’я користувача в системі	Nvarchar(256)	
NormalizerUserName	Ім’я користувача в системі у верхньому регістрі	Nvarchar(256)	
Email	Поштова адреса	Nvarchar(256)	
NormalizerEmail	Поштова адреса у верхньому регістрі	Nvarchar(256)	
EmailConfirmed	Підтвердження поштової адреси	Bit	
PasswordHash	Хеш паролю	Nvarchar(MAX)	
SecurityStamp	Мітка безпеки	Nvarchar(MAX)	
PhoneNumber	Номер телефону	Nvarchar(MAX)	

2.4 Аналіз безпеки даних

Забезпечення надійного збереження інформації в базі даних є важливим етапом при розробці ПЗ. В нашому програмному забезпеченні хешування відбувається за допомогою хеш-функції Sha256, яка має велику ступінь надійності.

На додачу до хеш-функції, також використовується HMAC(Hash-based message authentication code) - механізм перевірки цілісності інформації, який передбачає використання спільного секретного ключа для двох клієнтів.[7] Обидва ці рішення в парі гарантують високу степінь надійності даних.

Для підвищення надійності даних користувача, вони проходять етап валідації, який передбачає виконання наступних вимог:

- довжина паролю повинна бути не менше 8 символів;
- пароль повинен мати принаймні один символ у верхньому регістрі;
- пароль повинен мати принаймні один символ у нижньому регістрі;
- пароль повинен мати принаймні одну цифру;
- пароль не повинен мати спеціальних символів.

У разі, якщо введений пароль користувача не проходить валідацію, система повідомляє це.

Для зберігання сесії користувача на сайті були використані Cookie.

2.5 Структура бази даних

Для зберігання даних в нашому програмного забезпеченні було використано систему управління реляційними базами даних MS SQL Server.

Серед переваг цієї СУБД слід зазначити: срощене розгортання, передача та інтеграція великих даних, висока продуктивність, інтелектуальна обробка запитів та гарна безпека даних.[8]

На рисунку 2.15 зображена схема бази даних для здійснення бронювання готелю.

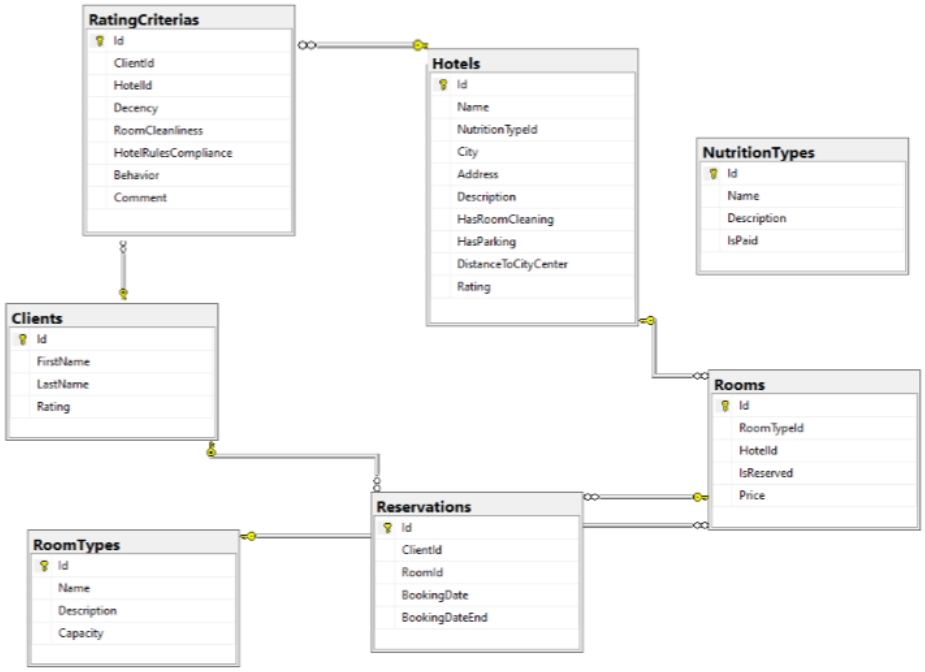


Рисунок 2.15 – Схема бази даних для здійснення бронювання готелю

Для створення бази даних для користувачів, було використано ASP .NET Core Identity. Це API, яке вбудоване в ASP .NET та підтримує функцію входу в клієнтську частину програмного забезпечення. Управляє користувачами, паролями, даними профілю, ролями, підтвердженням електронної пошти.[9]

На рисунку 2.16 зображена схема бази даних для збереження та управління даними користувачів нашого застосування.

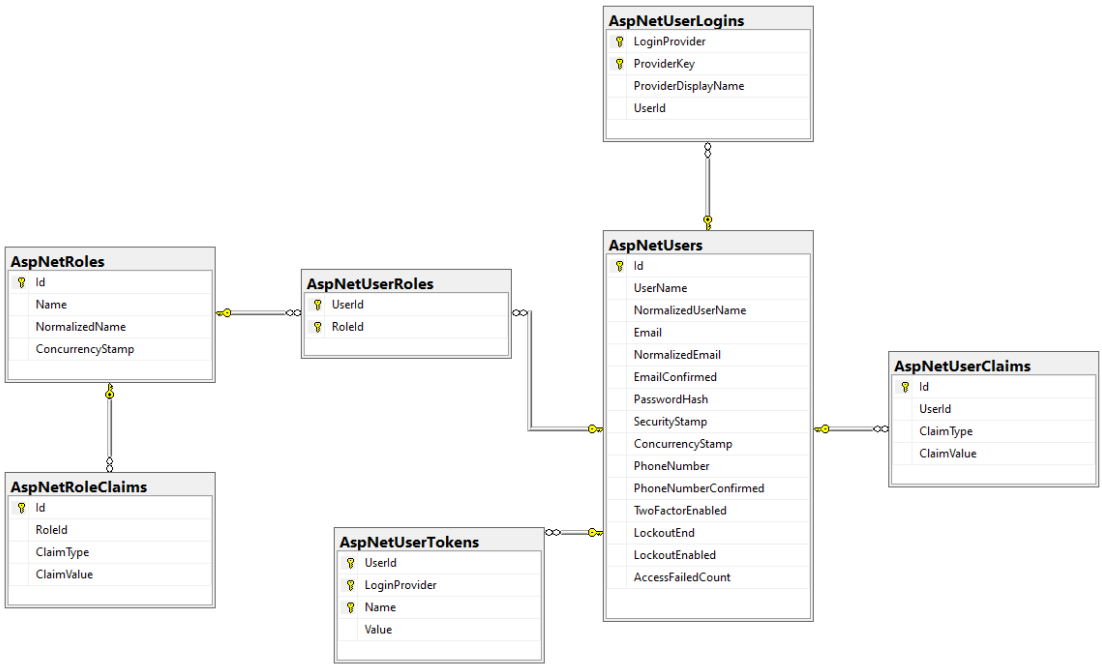


Рисунок 2.16 – Схема бази даних для збереження та управління даними користувачів

2.6 Висновки по розділу

В даному розділі було проведено моделювання та аналіз програмного забезпечення за допомогою BPMN діаграм бізнес-процесів. Було зроблено опис архітектури програмного забезпечення. Були описані основні класи, методи та контроллери.

Був проведений аналіз безпеки даних ПЗ, який показав високу степінь надійності.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування є важливим етапом розробки програмного забезпечення, бо гарантує відповідність технічному завданню, правильність функціонування написаного продукту та дозволяє виявити суттєві недоліки програмного коду.

Стратегія тестування - це процес аналізу програмного продукту для виявлення в ньому дефектів і багів, а також перевірки на відповідність вимог до програмного продукту.

Процеси тестування можуть бути поділені на ручні та автоматизовані. Останні набагато полегшують роботу програміста за рахунок гарантування, що зміни в коді, внесені програмістом, не зламають програму та не приведуть до припинення роботи вже існуючого функціоналу.

Існує велика кількість видів тестування, які використовуються програмістами при перевірці правильності продукту. Найвідоміші серед них:

- функціональне тестування(functional testing);
- системне тестування (system testing);
- тестування продуктивності(performance testing);
- регресивне тестування(regression testing);
- тестування безпеки(sequrity testing);
- модульне тестування(unit testing).

По закінченню тестування необхідно сформувати список дефектів, поділивши їх за пріорітетом на :

- критичні;
- помірні
- незначні.

Після формування списку дефектів необхідно сформувати висновок з чіткими кроками проведення подальшої розробки.

3.2 Опис процесів тестування

Процеси тестування можуть бути поділені на ручні та автоматизовані. Останні набагато полегшують роботу програміста за рахунок гарантування, що зміни в коді, внесені програмістом, не зламають програму та не приведуть до припинення роботи вже існуючого функціоналу.

Для перевірки якості та працездатності програмного продукту були розроблені тест-кейси, перераховані в таблиці 3.1

Таблиця 3.1 – Тести функціональності сервера

Ідентифікатор	Опис тест-кейсу
CF-1	Перевірити можливість реєстрації для неавторизованого користувача
CF-2	Перевірити неможливість реєстрації з неправильними даними
CF-3	Перевірити можливість реєстрації для авторизованого користувача
CF-4	Перевірити неможливість авторизації з неправильними даними
CF-5	Перевірити можливість авторизації з правильними даними
CF-6	Перевірити можливість реєстрації для авторизованого користувача
CF-7	Перевірити неможливість повторної авторизації користувача

Продовження таблиці 3.1

CF-8	Перевірити неможливість повторної реєстрації користувача
CF-9	Перевірити можливість переходу на сторінку з готелями користувачем
CF-10	Перевірити можливість переходу на сторінку з готелями адміністратором
CF-11	Перевірити можливість переходу на сторінку з готелями менеджером готелю
CF-12	Перевірити можливість фільтрації готелів користувачем
CF-13	Перевірити можливість фільтрації готелів адміністратором
CF-14	Перевірити можливість фільтрації готелів менеджером готелю
CF-15	Перевірити неможливість додавання критеріїв до фільтру користувачем
CF-16	Перевірити неможливість додавання критеріїв до фільтру менеджером готелю
CF-17	Перевірити можливість додавання критеріїв до фільтру адміністратором
CF-18	Перевірити неможливість додавання готелю користувачем
CF-19	Перевірити неможливість додавання готелю менеджером готелю
CF-20	Перевірити можливість додавання готелю адміністратором

Продовження таблиці 3.1

CF-21	Перевірити можливість додавання номерів готелю адміністратором
CF-22	Перевірити неможливість додавання номерів готелю користувачем
CF-23	Перевірити неможливість додавання номерів готелю менеджером готелю
CF-24	Перевірити неможливість бронювання з неправильно введеними даними користувачем
CF-25	Перевірити можливість бронювання з правильно введеними даними користувачем
CF-26	Перевірити неможливість бронювання без виконання оплати
CF-27	Перевірити можливість бронювання з оплатою
CF-28	Перевірити можливість виставлення рейтингу з відгуком користувачу менеджером готелю
CF-29	Перевірити неможливість виставлення рейтингу з відгуком користувачу іншим користувачем
CF-30	Перевірити неможливість виставлення рейтингу з відгуком користувачу адміністратором
CF-31	Перевірити можливість перегляду рейтингу користувача манеджером готелю

Продовження таблиці 3.1

CF-32	Перевірити неможливість перегляду рейтингу користувача іншим користувачем
CF-33	Перевірити можливість перегляду рейтингу користувача адміністратором
CF-34	Перевірити можливість користувача відправити текст менеджеру готелю у чаті на сайті
CF-35	Перевірити можливість менеджера готелю відправити текст користувачу у чаті на сайті

3.3 Звіт тестування

У таблиці 3.2 наведено звіт про результати тестування, що включає в себе всі тест кейзи, описані в таблиці 3.1.

Таблиця 3.2 – Звіт тестування

Ідентифікатор	Очікуваний результат	Фактичний результат	Статус
CF-1	Перевірити можливість реєстрації для неавторизованого користувача	Перевірити можливість реєстрації для неавторизованого користувача	Пройдено
CF-2	Перевірити неможливість	Перевірити неможливість	Пройдено

Продовження таблиці 3.2

	реєстрації з неправильними даними	реєстрації з неправильними даними	
CF-3	Перевірити можливість реєстрації для авторизованого користувача	Перевірити можливість реєстрації для авторизованого користувача	Пройдено
CF-4	Перевірити неможливість авторизації з неправильними даними	Перевірити неможливість авторизації з неправильними даними	Пройдено
CF-5	Перевірити можливість авторизації з правильними даними	Перевірити можливість авторизації з правильними даними	Пройдено
CF-6	Перевірити можливість реєстрації для авторизованого користувача	Перевірити можливість реєстрації для авторизованого користувача	Пройдено
CF-7	Перевірити неможливість повторної авторизації користувача	Перевірити неможливість повторної авторизації користувача	Пройдено

Продовження таблиці 3.2

CF-8	Перевірити неможливість повторної реєстрації користувача	Перевірити неможливість повторної реєстрації користувача	Пройдено
CF-9	Перевірити можливість переходу на сторінку з готелями користувачем	Перевірити можливість переходу на сторінку з готелями користувачем	Пройдено
CF-10	Перевірити можливість переходу на сторінку з готелями адміністратором	Перевірити можливість переходу на сторінку з готелями адміністратором	Пройдено
CF-11	Перевірити можливість переходу на сторінку з готелями менеджером готелю	Перевірити можливість переходу на сторінку з готелями менеджером готелю	Пройдено
CF-12	Перевірити можливість фільтрації готелів користувачем	Перевірити можливість фільтрації готелів користувачем	Пройдено
CF-13	Перевірити можливість фільтрації готелів адміністратором	Перевірити можливість фільтрації готелів адміністратором	Пройдено

Продовження таблиці 3.2

CF-14	Перевірити можливість фільтрації готелів менеджером готелю	Перевірити можливість фільтрації готелів менеджером готелю	Пройдено
CF-15	Перевірити неможливість додавання критеріїв до фільтру користувачем	Перевірити неможливість додавання критеріїв до фільтру користувачем	Пройдено
CF-16	Перевірити неможливість додавання критеріїв до фільтру менеджером готелю	Перевірити неможливість додавання критеріїв до фільтру менеджером готелю	Пройдено
CF-17	Перевірити можливість додавання критеріїв до фільтру адміністратором	Перевірити можливість додавання критеріїв до фільтру адміністратором	Пройдено
CF-18	Перевірити неможливість додавання готелю користувачем	Перевірити неможливість додавання готелю користувачем	Пройдено
CF-19	Перевірити неможливість додавання готелю менеджером готелю	Перевірити неможливість додавання готелю менеджером готелю	Пройдено

Продовження таблиці 3.2

CF-20	Перевірити можливість додавання готелю адміністратором	Перевірити можливість додавання готелю адміністратором	Пройдено
CF-21	Перевірити можливість додавання номерів готелю адміністратором	Перевірити можливість додавання номерів адміністратором	Пройдено
CF-22	Перевірити неможливість додавання номерів готелю користувачем	Перевірити неможливість додавання номерів готелю користувачем	Пройдено
CF-23	Перевірити неможливість додавання номерів готелю менеджером готелю	Перевірити неможливість додавання номерів готелю менеджером готелю	Пройдено
CF-24	Перевірити неможливість бронювання з неправильно введеними даними користувачем	Перевірити неможливість бронювання з неправильно введеними даними користувачем	Пройдено
CF-25	Перевірити можливість бронювання з правильно введеними даними користувачем	Перевірити можливість бронювання з правильно	Пройдено

Продовження таблиці 3.2

		введеними даними користувачем	
CF-26	Перевірити неможливість бронювання без виконання оплати	Перевірити неможливість бронювання без виконання оплати	Пройдено
CF-27	Перевірити можливість бронювання з оплатою	Перевірити можливість бронювання з оплатою	Пройдено
CF-28	Перевірити можливість виставлення рейтингу з відгуком користувачу менеджером готелю	Перевірити можливість виставлення рейтингу з відгуком користувачу менеджером готелю	Пройдено
CF-29	Перевірити неможливість виставлення рейтингу з відгуком користувачу іншим користувачем	Перевірити неможливість виставлення рейтингу з відгуком користувачу іншим користувачем	Пройдено
CF-30	Перевірити неможливість виставлення рейтингу з відгуком користувачу адміністратором	Неможливість виставлення рейтингу користувачу адміністратором	Пройдено

Продовження таблиці 3.2

CF-31	Перевірити можливість перегляду рейтингу користувача манеджером готелю	Перевірити можливість перегляду рейтингу користувача манеджером готелю	Пройдено
CF-32	Перевірити неможливість перегляду рейтингу користувача іншим користувачем	Перевірити неможливість перегляду рейтингу користувача іншим користувачем	Пройдено
CF-33	Перевірити можливість перегляду рейтингу користувача адміністратором	Перевірити можливість перегляду рейтингу користувача адміністратором	Пройдено
CF-34	Перевірити можливість користувача відправити текст менеджеру готелю у чаті на сайті	Перевірити можливість користувача відправити текст менеджеру готелю у чаті на сайті	Пройдено
CF-35	Перевірити можливість менеджера готелю відправити текст користувачу у чаті на сайті	Перевірити можливість менеджера готелю відправити текст користувачу у чаті на сайті	Пройдено

3.4 Висновки по розділу

В даному розділі було проведено аналіз якості нашого програмного забезпечення, визначено його оцінку та проведено тести функціональності сервера. Були описані цілі для успішного тестування веб-застосування, межі використання та основні складові елементи, що тестуються.

Також було продумано стратегію тестування програмного продукту та наведено контрольні приклади тестів. Послідовно були перевірені всі основні варіанти використання, результати були представлені у відповідних таблицях.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання даного програмного продукту необхідно використовувати:

– Kestrel, в якості серверу;

– базу даних MS SQL Server;

– .NET Core версії 3.0;

– Vue.js;

– Node.js;

– менеджер пакетів NPM.

Для коректної роботи фреймворку Vue.js необхідно через менеджер пакетів встановити додаткові бібліотеки Axios та Vuex для клієнт-серверного обміну інформацією.

Для роботи з програмним кодом рекомендується до використання Visual Studio 2017 та вище, а також Visual Studio Code для роботи з фреймворком Vue.js.

Послідовність запуску веб-застосування:

– запустити Kestrel;

– запустити сервер з Vue.js.[10]

4.2 Робота з програмним забезпеченням

Детально інструкцію з використання клієнтської частини програмного забезпечення наведено у документі «Керівництво користувача»

ВИСНОВКИ

Дипломний проєкт присвячений розробці веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів.

Мета розробки – створення веб-застосування для покращення процесів бронювання для користувачів та зменшення певних ризиків для бізнесу за рахунок рейтингової системи оцінювання клієнтів. Зі сторони користувачів це досягається за рахунок надання першим найпопулярніших пропозицій з найактуальнішою інформацією. Зі сторони бізнесу - це можливість оцінювання клієнта за певною рейтинговою системою, підтвердження оплат та можливість підтверджувати або скасовувати бронювання, якщо майбутній клієнт може спричинити певні ризики. Під ризиками розуміються збитки, які потенційно може понести бізнес, а саме: поганий настрій інших клієнтів, який негативно впливає на статус цього бізнесу, певні матеріальні втрати і, як наслідок, зниження загального рейтингу самого готелю.

У розділі «Аналіз вимог до програмного забезпечення» було сформульовано мету проведення розробки нашого веб-застосування, актуальність, основні задачі та цілі. Було проведене дослідження існуючих програмних продуктів, з'ясовано їх головні переваги та недоліки. Виконано аналіз відомих технічних рішень, вимог до програмного забезпечення. Сформовано перелік функціональних та нефункціональних вимог до програмного забезпечення.

У розділі «Моделювання та конструювання програмного забезпечення» було проведено моделювання та аналіз програмного забезпечення за допомогою BPMN діаграм бізнес-процесів. Було зроблено опис архітектури програмного забезпечення. Були описані основні класи репозиторіїв, сервісів та контроллерів. Був проведений аналіз безпеки даних ПЗ, який показав високу степінь надійності.

У розділі «Аналіз якості та тестування програмного забезпечення» було проведено аналіз якості нашого програмного забезпечення, визначено його оцінку та проведено тести функціональності сервера. Були описані цілі для успішного тестування веб-застосування, межі використання та основні складові елементи, що тестуються. Також було продумано стратегію тестування програмного продукту та наведено приклади тестів. Послідовно були перевірені всі основні варіанти використання, результати були представлені у відповідних таблицях.

У розділі «Впровадження та супровід програмного забечпечення» було вказано вимоги для розгортання серверної та клієнтської частини ПЗ, а інструкції для користувача були наведені в додатках.

ПЕРЕЛІК ПОСИЛАНЬ

1) Інформаційна система [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0

2) Какие есть типы номеров в отелях [Електронний ресурс] — Режим доступу: https://hotellook.ru/help/kakie-est-tipy-nomerov-v-otelyah

3) Booking.com [Електронний ресурс] — Режим доступу: https://www.booking.com/

4) Agoda.com [Електронний ресурс] — Режим доступу: https://www.agoda.com/uk-ua/

5) BPMN diagram [Електронний ресурс] — Режим доступу: https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation

6) N-tire architecture [Електронний ресурс] — Режим доступу: https://en.wikipedia.org/wiki/Multitier_architecture

7) HMAC [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/HMAC

8) Microsoft SQL Server Pros and Cons [Електронний ресурс] — Режим доступу: https://learnsql.com/blog/microsoft-sql-server-pros-and-cons/

9) Intoduction to Identity on ASP .NET Core [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio

10) Vue.js introduction [Електронний ресурс] — Режим доступу: https://vuejs.org/v2/guide/

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
“___” _____ 2020 р.

Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів

Технічне завдання

КПІ.ІП-6311.045440.02.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ К.І. Ліщук

Нормоконтроль:
_____ К.І. Ліщук

Виконавець:
_____Р.Л. Заєць

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів

Галузь застосування: Уся сфера готельного онлайн-бізнесу

Наведене технічне завдання поширюється на розробку веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів, котра використовується для надання можливості онлайн-бронювання готелів звичайним користувачам та зменшення певних ризиків для бізнесу.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» («КПІ ім. Ігоря Сікорського»).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для покращення процесу онлайн-бронювання готелів для користувачів та мінімізації потенційних ризиків для бізнесу на основі рейтингової системи оцінювання клієнтів

Метою розробки є спрощення процесів онлайн бронювання готелів для користувачів та запобігання виникнення потенційних ризиків для бізнесу, пов’язаних з некоректною поведінкою клієнтів.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

– реєстрація;
– атворизація;
– можливість перегляду доступних готелів;
– можливість фільтрації готелів за певними критеріями;
– можливість виконання бронювання номеру готелю ;
– можливість перегляду статусів бронювання номеру готелю;
– можливість виставлення рейтингової оцінки готелю, після перебування в ньому;

4.1.1.2 Для адміністратора:

– можливість ведення готелів;
– можливість ведення номерів готелів;
– можливість ведення пошукового фільтру;
– можливість оновлення важливої інформації на сайті.

4.1.1.3 Для менеджера готелю:

– можливість перегляду списку активних заявок на бронювання;
– можливість підтвердження сплат;
– можливість скасовувати або схвалювати бронювання;
– можливість перегляду рейтингової оцінки та відгуки про потенційного клієнта;

– можливість виставлення рейтингової оцінки та залишення відгуку для клієнту, після його перебування в готелі;

– можливість спілкування з клієнтом, використовуючи чат.

4.1.2 Розробку виконати на платформі Windows.

4.1.3 Додаткові вимоги:

– мова клієнтського інтерфейсу – англійська;
– перегляд сайту можливий лише для авторизованих користувачів;

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

При реєстрації користувач повинен ввести правильні дані, які задовольняють вимогами валідації, а саме: розмір паролю не менший за 8 символів, пароль повинен мати принаймні одну літеру у верхньому та нижньому регістрі, пароль повинен мати принаймні 1 цифру та не повинен мати спеціальних символів.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

Не висуваються

4.3.2 Обслуговування

Програмне забезпечення не вимагає специфічного обслуговування

4.3.3 Обслуговуючий персонал

Адміністратор повинен додавати, видаляти або оновлювати необхідні дані у базі даних.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Тип процесору Intel Core i3.

4.4.2.2 Об‘єм ОЗП 4 ГБ.

4.4.2.3 Об’єм жорсткого диску 100ГБ.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN64 (Windows 7, Windows 8 та Windows 10).

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: HTTP запити з можливими URL параметрами, та JSON значеннями у тілі запиту.

4.5.3 Результати повинні бути представлені в наступному форматі: Відповіді сервера на HTTP запити у форматі JSON.

4.5.4 Програмне забезпечення повинно обробляти HTTP-запити та відображати їх результати в браузері.

4.5.5 Програмне забезпечення повинно взаємодіяти з сервером за допомогою протоколу HTTPS.

4.5.6 Клієнтська частина повинна бути розроблена за допомогою фреймворку Vue.js та мов програмування CSS, HTML, серверна частина має бути створена з використанням технології ASP.NET Core 3.0 та мови програмування C#.

4.6. Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Сервер розгортається локально, окремо від клієнтської частини.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм верхнього рівня повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 60 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Опис програми.

5.3.4 Керівництво користувача.

5.4 Графічна частина повинна бути виконана на 3 аркушах формату А3.

5.4.1 Схема структурна варіантів використання.

5.4.2 Схема бази даних.

5.4.3 Креслення вигляду екранних форм.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	22.02.2020	
2.	Розробка технічного завдання	03.03.2020	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	18.03.2020	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	28.03.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму …)
5.	Програмна реалізація програмного забезпечення	04.04.2020	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.04.2020	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.04.2020	Пояснювальна записка.
8.	Розробка матеріалів графічної частини проекту	21.04.2020	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	27.04.2020	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до підрозділу «Випробування програмного продукту».

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ ДЛЯ БРОНЮВАННЯ ГОТЕЛІВ З ВИКОРИСТАННЯМ РЕЙТИНГОВОЇ СИСТЕМИ ОЦІНЮВАННЯ КЛІЄНТІВ

Опис програми

КПІ.ІП-6311.045440.03.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ К.І. Ліщук

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Р.Л. Заєць

Київ – 2020 року

Тексти програмного коду

Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

60 арк, 346 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020

1 КОНТРОЛЕРИ

```
public class AccountController : Controller
    {
        private readonly SignInManager<User> _signInManager;
        private readonly UserManager<User> _userManager;

        public AccountController(UserManager<User> userManager, SignInManager<User>
signInManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
        }

        [Route("api/account/register")]
        [HttpPost]
        public async Task<IActionResult> Register([FromBody]RegisterModel model)
        {
            if (!ModelState.IsValid)
            {
                return ValidationProblem();
            }
            if (ModelState.IsValid)
            {
                var user = new User { Email = model.Email, UserName = model.Name };
                var result = await _userManager.CreateAsync(user, model.Password);
                if (!result.Succeeded)
                {
                    return ValidationProblem();
                }
                if (result.Succeeded)
                {
                    await _signInManager.SignInAsync(user, false);
                }

                foreach (var error in result.Errors) ModelState.AddModelError(string.Empty,
error.Description);
            }


            return Ok(model);
        }

        [Route("api/account/login")]
        [HttpPost]
        public async Task<IActionResult> Login([FromBody]LoginModel model)
        {
            if (ModelState.IsValid)
            {
                var user = await _userManager.FindByEmailAsync(model.Email);
                var result = await _signInManager.PasswordSignInAsync(user, model.Password,
false, false);
                if (result.Succeeded)
                {
                    //TODO: Check for sub domain
                    if (!string.IsNullOrEmpty(model.ReturnUrl) /* &&
Url.IsLocalUrl(model.ReturnUrl)*/)
                        return Redirect(model.ReturnUrl);
                }

                ModelState.AddModelError("", "Wrong login or password");
            }

            return Ok(model);
        }
```

```
        [Route("api/account/register/confirm/{email}")]
        [HttpGet]
        public async Task<User> GetUserByEmail(string email)
        {
            var user = await _userManager.FindByEmailAsync(email.ToUpper());
            return user;
        }
    }

public class ClientsController : Controller
    {
        private readonly IClientService _clientsService;

        private readonly IMapper _mapper;

        public ClientsController(IClientService service, IMapper mapper)
        {
            _clientsService = service;
            _mapper = mapper;
        }

        [Route("api/clients/{id:int}")]
        [HttpGet]
        public async Task<IActionResult> Get(int id)
        {
            var client = await _clientsService.GetClientById(id);

            if (client == null)
            {
                return NotFound();
            }

            var clientView = _mapper.Map<ClientDTO, ClientView>(client);

            return Ok(clientView);
        }

        [Route("api/clients")]
        [HttpGet]
        public async Task<IEnumerable<ClientView>> GetAll()
        {
            var clients = await _clientsService.GetAllClients();

            return _mapper.Map<IEnumerable<ClientDTO>, IEnumerable<ClientView>>(clients);
        }

        [Route("api/clients")]
        [HttpPost]
        public async Task<IActionResult> Add([FromBody]ClientView client)
        {
            var clientDTO = _mapper.Map<ClientView, ClientDTO>(client);

            await _clientsService.AddClient(clientDTO);

            _clientsService.SaveChanges();

            return Ok();
        }

        [Route("api/clients")]
        [HttpPut]
        public async Task<IActionResult> Update([FromBody]ClientView client)
        {
            var clientDTO = _mapper.Map<ClientView, ClientDTO>(client);
```

```
            await _clientsService.UpdateClient(clientDTO);

            _clientsService.SaveChanges();

            return Ok();
        }

        [Route("api/clients/{id:int}")]
        [HttpDelete]
        public IActionResult Delete(int id)
        {
            if (_clientsService.DeleteClient(id))
            {
                _clientsService.SaveChanges();

                return Ok();
            }

            return NotFound();
        }

        [Route("api/clients/name/{clientFirstName}/{clientLastName}")]
        [HttpGet]
        public async Task<ClientView> GetClientByName(string clientFirstName, string
clientLastName)
        {
            var client = await _clientsService.GetClientByName(clientFirstName,
clientLastName);

            return _mapper.Map<ClientDTO, ClientView>(client);
        }
        [Route("api/clients/rating/{rating}")]
        [HttpGet]
        public async Task<IEnumerable<ClientView>> GetClientByRating(int rating)
        {
            var client = await _clientsService.GetClientsByRating(rating);

            return _mapper.Map<IEnumerable<ClientDTO>, IEnumerable<ClientView>>(client);
        }
    }

public class HotelsController : Controller
    {
        private readonly IHotelService _hotelsService;

        private readonly IMapper _mapper;

        public HotelsController(IHotelService service, IMapper mapper)
        {
            _hotelsService = service;
            _mapper = mapper;
        }

        [Route("api/hotels/{id:int}")]
        [HttpGet]
        public async Task<IActionResult> Get(int id)
        {
            var hotel = await _hotelsService.GetHotelById(id);

            if (hotel == null)
            {
                return NotFound();
            }
```

```
    var hotelView = _mapper.Map<HotelDTO, HotelView>(hotel);

    return Ok(hotelView);
}

[Route("api/hotels")]
[HttpGet]
public async Task<IEnumerable<HotelView>> GetAll()
{
    var hotels = await _hotelsService.GetAllHotels();

    return _mapper.Map<IEnumerable<HotelDTO>, IEnumerable<HotelView>>(hotels);
}

[Route("api/hotels")]
[HttpPost]
public async Task<IActionResult> Add([FromBody]HotelView hotel)
{
    var hotelDTO = _mapper.Map<HotelView, HotelDTO>(hotel);

    await _hotelsService.AddHotel(hotelDTO);

    _hotelsService.SaveChanges();

    return Ok();
}

[Route("api/hotels")]
[HttpPut]
public async Task<IActionResult> Update([FromBody]HotelView hotel)
{
    var hotelDTO = _mapper.Map<HotelView, HotelDTO>(hotel);

    await _hotelsService.UpdateHotel(hotelDTO);

    _hotelsService.SaveChanges();

    return Ok();
}

[Route("api/hotels/{id:int}")]
[HttpDelete]
public ActionResult Delete(int id)
{
    if (_hotelsService.DeleteHotel(id))
    {
        _hotelsService.SaveChanges();

        return Ok();
    }

    return NotFound();
}

[Route("api/hotels/name/{name}")]
[HttpGet]
public async Task<HotelView> GetHotelByName(string name)
{
    var hotels = await _hotelsService.GetHotelByName(name);

    return _mapper.Map<HotelDTO, HotelView>(hotels);
}
[Route("api/hotels/city/{city}")]
[HttpGet]
public async Task<IEnumerable<HotelView>> GetHotelsByCity(string city)
```

```
        {
            var hotels = await _hotelsService.GetHotelsByCity(city);

            return _mapper.Map<IEnumerable<HotelDTO>,IEnumerable<HotelView>>(hotels);
        }

        [Route("api/hotels/nutritionType/{nutritionTypeId}")]
        [HttpGet]
        public async Task<IEnumerable<HotelView>> GetHotelsByNutrition(int nutritionTypeId)
        {
            var hotels = await _hotelsService.GetHotelsByNutrition(nutritionTypeId);

            return _mapper.Map<IEnumerable<HotelDTO>, IEnumerable<HotelView>>(hotels);
        }

        [Route("api/hotels/rating/{rating}")]
        [HttpGet]
        public async Task<IEnumerable<HotelView>> GetHotelByRating(int rating)
        {
            var hotels = await _hotelsService.GetHotelsByRating(rating);

            return _mapper.Map<IEnumerable<HotelDTO>, IEnumerable<HotelView>>(hotels);
        }

        [Route("api/hotels/roomCleaning/{hasRoomCleaning}")]
        [HttpGet]
        public async Task<IEnumerable<HotelView>> GetHotelByRoomCleaning(bool
hasRoomCleaning)
        {
            var hotels = await _hotelsService.GetHotelsByRoomCleaning(hasRoomCleaning);

            return _mapper.Map<IEnumerable<HotelDTO>, IEnumerable<HotelView>>(hotels);
        }

        [Route("api/hotels/parking/{hasParking}")]
        [HttpGet]
        public async Task<IEnumerable<HotelView>> GetHotelByParking(bool hasParking)
        {
            var hotels = await _hotelsService.GetHotelsByParking(hasParking);

            return _mapper.Map<IEnumerable<HotelDTO>, IEnumerable<HotelView>>(hotels);
        }

        [Route("api/hotels/ratingArray/{rat1}/{rat2}/{rat3}")]
        [HttpGet]
        public async Task<IEnumerable<HotelView>> GetHotelsByRatingArray(int rat1, int rat2,
int rat3)
        {
            var hotels = await _hotelsService.GetHotelsByRatingArray(rat1, rat2, rat3);

            return _mapper.Map<IEnumerable<HotelDTO>, IEnumerable<HotelView>>(hotels);
        }

        [Route("api/hotels/distance/{dist1}/{dist2}/{dist3}")]
        [HttpGet]
        public async Task<IEnumerable<HotelView>> GetHotelsByDistanceArray(double dist1,
double dist2, double dist3)
        {
            var hotels = await _hotelsService.GetHotelsByDistanceArray(dist1, dist2, dist3);

            return _mapper.Map<IEnumerable<HotelDTO>, IEnumerable<HotelView>>(hotels);
        }

        [Route("api/hotels/nutritionArray/{ro}/{bb}/{hb}/{fb}/{ai}")]
        [HttpGet]
```

```
        public async Task<IEnumerable<HotelView>> GetHotelsByNutritionArray(int ro, int bb,
int hb, int fb, int ai)
        {
            var hotels = await _hotelsService.GetHotelsByNutritionArray(ro, bb, hb,fb,ai);

            return _mapper.Map<IEnumerable<HotelDTO>, IEnumerable<HotelView>>(hotels);
        }
    }

public class NutritionController : ControllerBase
    {
        private readonly INutritionTypeService _nutritionTypeService;

        private readonly IMapper _mapper;

        public NutritionController(INutritionTypeService service, IMapper mapper)
        {
            _nutritionTypeService = service;
            _mapper = mapper;
        }

        [Route("api/nutritionTypes/{id:int}")]
        [HttpGet]
        public async Task<IActionResult> Get(int id)
        {
            var nutritionType = await _nutritionTypeService.GetNutritionTypeById(id);

            if (nutritionType == null)
            {
                return NotFound();
            }

            var nutritionTypeView = _mapper.Map<NutritionTypeDTO,
NutritionTypeView>(nutritionType);

            return Ok(nutritionTypeView);
        }

        [Route("api/nutritionTypes")]
        [HttpGet]
        public async Task<IEnumerable<NutritionTypeView>> GetAll()
        {
            var nutritionTypes = await _nutritionTypeService.GetAllNutritionTypes();

            return _mapper.Map<IEnumerable<NutritionTypeDTO>,
IEnumerable<NutritionTypeView>>(nutritionTypes);
        }


        [Route("api/nutritionTypes")]
        [HttpPost]
        public async Task<IActionResult> Add([FromBody]NutritionTypeView nutritionType)
        {
            var nutritionTypesDTO = _mapper.Map<NutritionTypeView,
NutritionTypeDTO>(nutritionType);

            await _nutritionTypeService.AddNutritionType(nutritionTypesDTO);

            _nutritionTypeService.SaveChanges();

            return Ok();
        }

        [Route("api/nutritionTypes")]
        [HttpPut]
```

```
        public async Task<IActionResult> Update([FromBody]NutritionTypeView nutritionType)
        {
            var nutritionTypeDTo = _mapper.Map<NutritionTypeView,
NutritionTypeDTO>(nutritionType);

            await _nutritionTypeService.UpdateNutritionType(nutritionTypeDTo);

            _nutritionTypeService.SaveChanges();

            return Ok();
        }

        [Route("api/nutritionTypes/{id:int}")]
        [HttpDelete]
        public IActionResult Delete(int id)
        {
            if (_nutritionTypeService.DeleteNutritionType(id))
            {
                _nutritionTypeService.SaveChanges();
                return Ok();
            }

            return NotFound();
        }
    }

public class RatingCriteriasController : Controller
    {
        private readonly IRatingCriteriaService _ratingCriteriaService;

        private readonly IMapper _mapper;

        public RatingCriteriasController(IRatingCriteriaService service, IMapper mapper)
        {
            _ratingCriteriaService = service;
            _mapper = mapper;
        }

        [Route("api/rating")]
        [HttpPost]
        public async Task<IActionResult> AddRating(RatingCriteriaView ratingCriteria)
        {
            var ratingCriteriaDTO = _mapper.Map<RatingCriteriaView,
RatingCriteriaDTO>(ratingCriteria);

            await _ratingCriteriaService.AddRating(ratingCriteriaDTO);

            _ratingCriteriaService.SaveChanges();

            return Ok();
        }

        [Route("api/rating")]
        [HttpPut]
        public async Task<IActionResult> Update([FromBody]RatingCriteriaView rating)
        {
            var ratingDTO = _mapper.Map<RatingCriteriaView, RatingCriteriaDTO>(rating);

            await _ratingCriteriaService.UpdateRating(ratingDTO);

            _ratingCriteriaService.SaveChanges();

            return Ok();
        }
```

```
        [Route("api/rating/{id:int}")]
        [HttpDelete]
        public IActionResult Delete(int id)
        {
            if (_ratingCriteriaService.DeleteRating(id))
            {
                _ratingCriteriaService.SaveChanges();

                return Ok();
            }

            return NotFound();
        }

        [Route("api/rating/{id:int}")]
        [HttpGet]
        public async Task<IActionResult> GetRatingById(int ratingCriteriaId)
        {
            var rating = await _ratingCriteriaService.GetRatingById(ratingCriteriaId);

            if (rating == null)
            {
                return NotFound();
            }

            var roomView = _mapper.Map<RatingCriteriaDTO, RatingCriteriaView>(rating);

            return Ok(roomView);
        }

        [Route("api/rating/")]
        [HttpGet]
        public async Task<IEnumerable<RatingCriteriaView>> GetAllRatings()
        {

            var rating = await _ratingCriteriaService.GetAllRatings();

            return _mapper.Map<IEnumerable<RatingCriteriaDTO>,
IEnumerable<RatingCriteriaView>>(rating);
        }

        [Route("api/rating/clients/{clientId:int}")]
        [HttpGet]
        public async Task<IEnumerable<RatingCriteriaView>> GetRatingByClientId(int clientId)
        {

            var rating = await _ratingCriteriaService.GetRatingByClientId(clientId);

            return _mapper.Map<IEnumerable<RatingCriteriaDTO>,
IEnumerable<RatingCriteriaView>>(rating);
        }

        [Route("api/rating/hotels/{hotelId:int}")]
        [HttpGet]
        public async Task<IEnumerable<RatingCriteriaView>> GetRatingByByHotelId(int hotelId)
        {
            var rating = await _ratingCriteriaService.GetRatingByByHotelId(hotelId);

            return _mapper.Map<IEnumerable<RatingCriteriaDTO>,
IEnumerable<RatingCriteriaView>>(rating);
        }
    }

public class ReservationsController : Controller
    {
```

```
        private readonly IReservationService _reservationsService;

        private readonly IMapper _mapper;

        public ReservationsController(IReservationService service, IMapper mapper)
        {
            _reservationsService = service;
            _mapper = mapper;
        }

        [Route("api/reservations/{id:int}")]
        [HttpGet]
        public async Task<IActionResult> Get(int id)
        {
            var reservation = await _reservationsService.GetReservationById(id);

            if (reservation == null)
            {
                return NotFound();
            }

            var reservationView = _mapper.Map<ReservationDTO, ReservationView>(reservation);

            return Ok(reservationView);
        }
        [Route("api/reservations")]
        [HttpGet]
        public async Task<IEnumerable<ReservationView>> GetAll()
        {
            var reservations = await _reservationsService.GetAllReservations();

            return _mapper.Map<IEnumerable<ReservationDTO>,
IEnumerable<ReservationView>>(reservations);
        }

        [Route("api/reservations")]
        [HttpPost]
        public async Task<IActionResult> Add([FromBody]ReservationView reservation)
        {
            var reservationDTO = _mapper.Map<ReservationView, ReservationDTO>(reservation);

            await _reservationsService.AddReservation(reservationDTO);

            _reservationsService.SaveChanges();

            return Ok();
        }

        [Route("api/reservations")]
        [HttpPut]
        public async Task<IActionResult> Update([FromBody]ReservationView reservation)
        {
            var reservationDTO = _mapper.Map<ReservationView, ReservationDTO>(reservation);

            await _reservationsService.UpdateReservation(reservationDTO);

            _reservationsService.SaveChanges();

            return Ok();
        }

        [Route("api/reservations/{id:int}")]
        [HttpDelete]
        public IActionResult Delete(int id)
        {
```

```
            if (_reservationsService.DeleteReservation(id))
            {
                _reservationsService.SaveChanges();

                return Ok();
            }

            return NotFound();
        }

        [Route("api/reservations/dateInterval/{bookingDate}/{bookingDateEnd}")]
        [HttpGet]
        public async Task<IEnumerable<ReservationView>> GetReservationByDate(DateTime
bookingDate, DateTime bookingDateEnd)
        {
            var reservations = await _reservationsService.GetReservationsByDate(bookingDate,
bookingDateEnd);

            return _mapper.Map<IEnumerable<ReservationDTO>,
IEnumerable<ReservationView>>(reservations);
        }
        [Route("api/reservations/{reservationId:int}/client")]
        [HttpGet]
        public async Task<ClientView> GetClientByReservation(int reservationId)
        {
            var client = await _reservationsService.GetClientByReservation(reservationId);

            return _mapper.Map<ClientDTO, ClientView>(client);
        }

        [Route("api/hotels/{hotelId:int}/clients")]
        [HttpGet]
        public async Task<IEnumerable<ClientView>> GetClientByHotel(int hotelId)
        {
            var clients = await _reservationsService.GetClientsByHotel(hotelId);

            return _mapper.Map<IEnumerable<ClientDTO>, IEnumerable<ClientView>>(clients);
        }

        [Route("api/reservations/{reservationId:int}/room")]
        [HttpGet]
        public async Task<RoomView> GetRoomByReservation(int reservationId)
        {
            var room = await _reservationsService.GetRoomByReservation(reservationId);

            return _mapper.Map<RoomDTO, RoomView>(room);
        }

    }

public class RoomsController : Controller
    {
        private readonly IRoomService _roomsService;

        private readonly IMapper _mapper;

        public RoomsController(IRoomService service, IMapper mapper)
        {
            _roomsService = service;
            _mapper = mapper;
        }

        [Route("api/rooms/{id:int}")]
        [HttpGet]
        public async Task<IActionResult> Get(int id)
```

```
{
    var room = await _roomsService.GetRoomById(id);

    if (room == null)
    {
        return NotFound();
    }

    var roomView = _mapper.Map<RoomDTO, RoomView>(room);

    return Ok(roomView);
}

[Route("api/rooms")]
[HttpPost]
public async Task<IActionResult> Add([FromBody]RoomView room)
{
    var roomDTO = _mapper.Map<RoomView, RoomDTO>(room);

    await _roomsService.AddRoom(roomDTO);

    _roomsService.SaveChanges();

    return Ok();
}

[Route("api/rooms")]
[HttpPut]
public async Task<IActionResult> Update([FromBody]RoomView room)
{
    var roomDTO = _mapper.Map<RoomView, RoomDTO>(room);

    await _roomsService.UpdateRoom(roomDTO);

    _roomsService.SaveChanges();

    return Ok();
}

[Route("api/rooms/{id:int}")]
[HttpDelete]
public IActionResult Delete(int id)
{
    if (_roomsService.DeleteRoom(id))
    {
        _roomsService.SaveChanges();

        return Ok();
    }

    return NotFound();
}

[Route("api/rooms/hotel/{hotelId:int}")]
[HttpGet]
public async Task<IEnumerable<RoomView>> GetRoomsByHotel(int hotelId)
{
    var rooms = await _roomsService.GetRoomsByHotel(hotelId);

    return _mapper.Map<IEnumerable<RoomDTO>, IEnumerable<RoomView>>(rooms);
}
[Route("api/rooms/roomType/{roomTypeId:int}")]
[HttpGet]
public async Task<IEnumerable<RoomView>> GetRoomsByRoomType(int roomTypeId)
{
```

```
            var rooms = await _roomsService.GetRoomsByRoomType(roomTypeId);

            return _mapper.Map<IEnumerable<RoomDTO>, IEnumerable<RoomView>>(rooms);
        }
    }

public class RoomTypesController : Controller
    {
        private readonly IRoomTypeService _roomTypesService;

        private readonly IMapper _mapper;

        public RoomTypesController(IRoomTypeService service, IMapper mapper)
        {
            _roomTypesService = service;
            _mapper = mapper;
        }

        [Route("api/roomTypes/{id:int}")]
        [HttpGet]
        public async Task<IActionResult> Get(int id)
        {
            var roomType = await _roomTypesService.GetRoomTypeById(id);

            if (roomType == null)
            {
                return NotFound();
            }

            var roomTypeView = _mapper.Map<RoomTypeDTO, RoomTypeView>(roomType);

            return Ok(roomTypeView);
        }

        [Route("api/roomTypes")]
        [HttpGet]
        public async Task<IEnumerable<RoomTypeView>> GetAll()
        {
            var roomTypes = await _roomTypesService.GetAllRoomTypes();

            return _mapper.Map<IEnumerable<RoomTypeDTO>,
IEnumerable<RoomTypeView>>(roomTypes);
        }

        [Route("api/roomTypes")]
        [HttpPost]
        public async Task<IActionResult> Add([FromBody]RoomTypeView roomType)
        {
            var roomTypeDTO = _mapper.Map<RoomTypeView, RoomTypeDTO>(roomType);

            await _roomTypesService.AddRoomType(roomTypeDTO);

            _roomTypesService.SaveChanges();

            return Ok();
        }

        [Route("api/roomTypes")]
        [HttpPut]
        public async Task<IActionResult> Update([FromBody]RoomTypeView roomType)
        {
            var roomTypeDTO = _mapper.Map<RoomTypeView, RoomTypeDTO>(roomType);

            await _roomTypesService.UpdateRoomType(roomTypeDTO);
```

```
            _roomTypesService.SaveChanges();

            return Ok();
        }

        [Route("api/roomTypes/{id:int}")]
        [HttpDelete]
        public IActionResult Delete(int id)
        {
            if (_roomTypesService.DeleteRoomType(id))
            {
                _roomTypesService.SaveChanges();

                return Ok();
            }

            return NotFound();
        }
    }
```

2 СЕРВІСИ

```
public class ClientService : IClientService
    {
        private readonly IUnitOfWork _uow;

        private readonly IMapper _mapper;
        public ClientService(IUnitOfWork uow, IMapper mapper)
        {
            _uow = uow;
            _mapper = mapper;
        }

        public async Task<ClientDTO> GetClientById(int clientId)
        {
            var client = await _uow.Clients.Get(clientId);
            return _mapper.Map<Client, ClientDTO>(client);
        }

        public async Task<IEnumerable<ClientDTO>> GetAllClients()
        {
            var clients = await _uow.Clients.GetAll();
            return _mapper.Map<IEnumerable<Client>, IEnumerable<ClientDTO>>(clients);
        }
        public async Task AddClient(ClientDTO client)
        {
            await _uow.Clients.Create(_mapper.Map<ClientDTO, Client>(client));
        }
        public async Task UpdateClient(ClientDTO client)
        {
            await _uow.Clients.Update(_mapper.Map<ClientDTO, Client>(client));
        }

        public bool DeleteClient(int clientId)
        {
            return _uow.Clients.Delete(clientId);
        }

        public async Task<ClientDTO> GetClientByName(string clientFirstName, string
clientLastname)
        {
            var client = (await _uow.Clients.Find(c => c.FirstName == clientFirstName &&
c.LastName == clientLastname)).First();
```

```
            return _mapper.Map<Client, ClientDTO>(client);
        }

        public async Task<IEnumerable<ClientDTO>> GetClientsByRating(double rating)
        {
            var clients = await _uow.Clients.Find(c => c.Rating == rating);
            return _mapper.Map<IEnumerable<Client>, IEnumerable<ClientDTO>>(clients);
        }
        public void SaveChanges()
        {
            _uow.Save();
        }

        public void Dispose()
        {
            _uow.Dispose();
        }
    }

public class HotelService : IHotelService
    {
        private readonly IUnitOfWork _uow;

        private readonly IMapper _mapper;

        public HotelService(IUnitOfWork uow, IMapper mapper)
        {
            _uow = uow;
            _mapper = mapper;
        }

        public async Task<HotelDTO> GetHotelById(int hotelId)
        {
            var hotel = await _uow.Hotels.Get(hotelId);
            return _mapper.Map<Hotel, HotelDTO>(hotel);
        }

        public async Task<IEnumerable<HotelDTO>> GetAllHotels()
        {
            var hotels = await _uow.Hotels.GetAll();
            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotels);
        }
        public async Task AddHotel(HotelDTO hotel)
        {
            await _uow.Hotels.Create(_mapper.Map<HotelDTO, Hotel>(hotel));
        }
        public async Task UpdateHotel(HotelDTO hotel)
        {
            await _uow.Hotels.Update(_mapper.Map<HotelDTO, Hotel>(hotel));
        }

        public bool DeleteHotel(int hotelId)
        {
            return _uow.Hotels.Delete(hotelId);
        }

        public async Task<HotelDTO> GetHotelByName(string hotelName)
        {
            var hotel = (await _uow.Hotels.Find(h => h.Name == hotelName)).First();
            return _mapper.Map<Hotel, HotelDTO>(hotel);
        }

        public async Task<IEnumerable<HotelDTO>> GetHotelsByCity(string city)
        {
            var hotels = await _uow.Hotels.Find(h => h.City == city);
```

```
            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotels);
        }

        public async Task<IEnumerable<HotelDTO>> GetHotelsByNutrition(int nutritionTypeId)
        {
            var hotels = await _uow.Hotels.Find(h => h.NutritionTypeId == nutritionTypeId);
            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotels);
        }

        public async Task<IEnumerable<HotelDTO>> GetHotelsByRating(int rating)
        {
            var hotels = await _uow.Hotels.Find(h => h.Rating == rating);
            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotels);
        }
        public void SaveChanges()
        {
            _uow.Save();
        }

        public void Dispose()
        {
            _uow.Dispose();
        }

        public async Task<IEnumerable<HotelDTO>> GetHotelsByRoomCleaning(bool
hasRoomCleaning)
        {
            var hotels = await _uow.Hotels.Find(h => h.HasRoomCleaning == hasRoomCleaning);
            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotels);
        }

        public async Task<IEnumerable<HotelDTO>> GetHotelsByParking(bool hasParking)
        {
            var hotels = await _uow.Hotels.Find(h => h.HasParking == hasParking);
            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotels);
        }

        public async Task<IEnumerable<HotelDTO>> GetHotelsByRatingArray(int rat1, int rat2,
int rat3)
        {
            var hotels = await _uow.Hotels.Find(h => h.Rating == rat1 || h.Rating == rat2 ||
h.Rating == rat3);
            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotels);
        }

        public async Task<IEnumerable<HotelDTO>> GetHotelsByDistanceArray(double dist1,
double dist2, double dist3)
        {
            var hotelsList = new List<Hotel>();
            IEnumerable<Hotel> hotels = null;
            if (dist1 != 0)
            {
                hotels = await _uow.Hotels.Find(h => h.DistanceToCityCenter < dist1);
                hotelsList.AddRange(hotels);
            }
            if(dist2 != 0)
            {
                hotels = await _uow.Hotels.Find(h => h.DistanceToCityCenter >= 1 &&
h.DistanceToCityCenter <= dist2);
                hotelsList.AddRange(hotels);
            }
            if(dist3 != 0)
            {
                hotels = await _uow.Hotels.Find(h => h.DistanceToCityCenter > dist3);
                hotelsList.AddRange(hotels);
```

```
            }

            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotelsList);
        }

        public async Task<IEnumerable<HotelDTO>> GetHotelsByNutritionArray(int ro, int bb,
int hb, int fb, int ai)
        {
            var hotels = await _uow.Hotels.Find(h => h.NutritionTypeId == ro ||
h.NutritionTypeId == bb || h.NutritionTypeId == hb || h.NutritionTypeId == fb ||
h.NutritionTypeId == ai);
            return _mapper.Map<IEnumerable<Hotel>, IEnumerable<HotelDTO>>(hotels);
        }
    }

public class NutritionTypeService : INutritionTypeService
    {
        private readonly IUnitOfWork _uow;

        private readonly IMapper _mapper;
        public NutritionTypeService(IUnitOfWork uow, IMapper mapper)
        {
            _uow = uow;
            _mapper = mapper;
        }
        public async Task AddNutritionType(NutritionTypeDTO nutritionType)
        {
            await _uow.NutritionTypes.Create(_mapper.Map<NutritionTypeDTO,
NutritionType>(nutritionType));
        }

        public bool DeleteNutritionType(int nutritionTypeId)
        {
            return _uow.NutritionTypes.Delete(nutritionTypeId);
        }

        public void Dispose()
        {
            _uow.Dispose();
        }

        public async Task<IEnumerable<NutritionTypeDTO>> GetAllNutritionTypes()
        {
            var nutritionTypes = await _uow.NutritionTypes.GetAll();
            return _mapper.Map<IEnumerable<NutritionType>,
IEnumerable<NutritionTypeDTO>>(nutritionTypes);
        }

        public async Task<NutritionTypeDTO> GetNutritionTypeById(int nutritionTypeId)
        {
            var nutritionType = await _uow.NutritionTypes.Get(nutritionTypeId);

            return _mapper.Map<NutritionType, NutritionTypeDTO>(nutritionType);
        }

        public void SaveChanges()
        {
            _uow.Save();
        }

        public async Task UpdateNutritionType(NutritionTypeDTO nutritionType)
        {
            await _uow.NutritionTypes.Update(_mapper.Map<NutritionTypeDTO,
NutritionType>(nutritionType));
        }
```

```
    }

public class RatingCriteriaService : IRatingCriteriaService
    {
        private readonly IUnitOfWork _uow;

        private readonly IMapper _mapper;
        public RatingCriteriaService(IUnitOfWork uow, IMapper mapper)
        {
            _uow = uow;
            _mapper = mapper;
        }
        public async Task AddRating(RatingCriteriaDTO ratingCriteria)
        {
            await _uow.RatingCriterias.Create(_mapper.Map<RatingCriteriaDTO,
RatingCriteria>(ratingCriteria));
        }

        public bool DeleteRating(int ratingId)
        {
            return _uow.RatingCriterias.Delete(ratingId);
        }

        public void Dispose()
        {
            _uow.Dispose();
        }

        public async Task<IEnumerable<RatingCriteriaDTO>> GetAllRatings()
        {
            var ratings = await _uow.RatingCriterias.GetAll();
            return _mapper.Map<IEnumerable<RatingCriteria>,
IEnumerable<RatingCriteriaDTO>>(ratings);
        }

        public async Task<IEnumerable<RatingCriteriaDTO>> GetRatingByByHotelId(int hotelId)
        {
            var ratings = await _uow.RatingCriterias.Find(r => r.HotelId == hotelId);
            return _mapper.Map<IEnumerable<RatingCriteria>,
IEnumerable<RatingCriteriaDTO>>(ratings);
        }

        public async Task<IEnumerable<RatingCriteriaDTO>> GetRatingByClientId(int clientId)
        {
            var ratings = await _uow.RatingCriterias.Find(r => r.ClientId == clientId);
            return _mapper.Map<IEnumerable<RatingCriteria>,
IEnumerable<RatingCriteriaDTO>>(ratings);
        }

        public async Task<RatingCriteriaDTO> GetRatingById(int ratingCriteriaId)
        {
            var rating = await _uow.RatingCriterias.Get(ratingCriteriaId);
            return _mapper.Map<RatingCriteria, RatingCriteriaDTO>(rating);
        }

        public void SaveChanges()
        {
            _uow.Save();
        }

        public async Task UpdateRating(RatingCriteriaDTO ratingCriteria)
        {
            await _uow.RatingCriterias.Update(_mapper.Map<RatingCriteriaDTO,
RatingCriteria>(ratingCriteria));
        }
```

```
    }

public class ReservationService : IReservationService
    {
        private readonly IUnitOfWork _uow;

        private readonly IMapper _mapper;

        public ReservationService(IUnitOfWork uow, IMapper mapper)
        {
            _uow = uow;
            _mapper = mapper;
        }

        public async Task<ReservationDTO> GetReservationById(int reservationId)
        {
            var reservation = await _uow.Reservations.Get(reservationId);
            return _mapper.Map<Reservation, ReservationDTO>(reservation);
        }

        public async Task<IEnumerable<ReservationDTO>> GetAllReservations()
        {
            var reservations = await _uow.Reservations.GetAll();
            return _mapper.Map<IEnumerable<Reservation>,
IEnumerable<ReservationDTO>>(reservations);
        }

        public async Task AddReservation(ReservationDTO reservation)
        {
            await _uow.Reservations.Create(_mapper.Map<ReservationDTO,
Reservation>(reservation));
        }

        public async Task UpdateReservation(ReservationDTO reservation)
        {
            await _uow.Reservations.Update(_mapper.Map<ReservationDTO,
Reservation>(reservation));
        }

        public bool DeleteReservation(int reservationId)
        {
            return _uow.Reservations.Delete(reservationId);
        }

        public async Task<IEnumerable<ReservationDTO>> GetReservationsByDate(DateTime
bookingDate, DateTime bookingDateEnd)
        {
            var reservations =
                await _uow.Reservations.Find(r => r.BookingDate >= bookingDate &&
r.BookingDateEnd <= bookingDateEnd);
            return _mapper.Map<IEnumerable<Reservation>,
IEnumerable<ReservationDTO>>(reservations);
        }

        public async Task<ClientDTO> GetClientByReservation(int reservationId)
        {
            var client = await _uow.Reservations.GetClientByReservation(reservationId);
            return _mapper.Map<Client, ClientDTO>(client);
        }

        public async Task<IEnumerable<ClientDTO>> GetClientsByHotel(int hotelId)
        {
            var clients = await _uow.Reservations.GetClientsByHotel(hotelId);
            return _mapper.Map<IEnumerable<Client>, IEnumerable<ClientDTO>>(clients);
        }
```

```
        public async Task<RoomDTO> GetRoomByReservation(int reservationId)
        {
            var room = await _uow.Reservations.GetRoomByReservation(reservationId);
            return _mapper.Map<Room, RoomDTO>(room);
        }
        public void SaveChanges()
        {
            _uow.Save();
        }

        public void Dispose()
        {
            _uow.Dispose();
        }
    }

public class RoomService : IRoomService
    {
        private readonly IUnitOfWork _uow;

        private readonly IMapper _mapper;

        public RoomService(IUnitOfWork uow, IMapper mapper)
        {
            _uow = uow;
            _mapper = mapper;
        }

        public async Task<RoomDTO> GetRoomById(int roomId)
        {
            var room = await _uow.Rooms.Get(roomId);

            return _mapper.Map<Room, RoomDTO>(room);
        }

        public async Task<IEnumerable<RoomDTO>> GetRoomsByHotel(int hotelId)
        {
            var rooms = await _uow.Rooms.Find(r => r.HotelId == hotelId);
            return _mapper.Map<IEnumerable<Room>, IEnumerable<RoomDTO>>(rooms);

        }

        public async Task<IEnumerable<RoomDTO>> GetRoomsByRoomType(int roomTypeId)
        {
            var rooms = await _uow.Rooms.Find(r => r.RoomTypeId == roomTypeId);
            return _mapper.Map<IEnumerable<Room>, IEnumerable<RoomDTO>>(rooms);
        }
        public async Task AddRoom(RoomDTO room)
        {
            await _uow.Rooms.Create(_mapper.Map<RoomDTO, Room>(room));
        }
        public async Task UpdateRoom(RoomDTO room)
        {
            await _uow.Rooms.Update(_mapper.Map<RoomDTO, Room>(room));
        }

        public bool DeleteRoom(int roomId)
        {
            return _uow.Rooms.Delete(roomId);
        }
        public void SaveChanges()
        {
            _uow.Save();
        }
```

```
        public void Dispose()
        {
            _uow.Dispose();
        }
    }

public class RoomTypeService : IRoomTypeService
    {
        private readonly IUnitOfWork _uow;

        private readonly IMapper _mapper;

        public RoomTypeService(IUnitOfWork uow, IMapper mapper)
        {
            _uow = uow;
            _mapper = mapper;
        }
        public async Task<RoomTypeDTO> GetRoomTypeById(int roomTypeId)
        {
            var roomType = await _uow.RoomTypes.Get(roomTypeId);

            return _mapper.Map<RoomType, RoomTypeDTO>(roomType);
        }

        public async Task<IEnumerable<RoomTypeDTO>> GetAllRoomTypes()
        {
            var roomTypes = await _uow.RoomTypes.GetAll();

            return _mapper.Map<IEnumerable<RoomType>, IEnumerable<RoomTypeDTO>>(roomTypes);
        }

        public async Task AddRoomType(RoomTypeDTO roomType)
        {
            await _uow.RoomTypes.Create(_mapper.Map<RoomTypeDTO, RoomType>(roomType));
        }
        public async Task UpdateRoomType(RoomTypeDTO roomType)
        {
            await _uow.RoomTypes.Update(_mapper.Map<RoomTypeDTO, RoomType>(roomType));
        }

        public bool DeleteRoomType(int roomTypeId)
        {
            return _uow.RoomTypes.Delete(roomTypeId);
        }
        public void SaveChanges()
        {
            _uow.Save();
        }

        public void Dispose()
        {
            _uow.Dispose();
        }
    }
```

3 РЕПОЗИТОРІЇ

```
public class ClientRepository : IRepository<Client>
    {
        private readonly HotelContext _db;

        public ClientRepository(HotelContext db)
        {
            _db = db;
        }

        public async Task<IEnumerable<Client>> GetAll()
        {
            return _db.Clients;
        }

        public async Task<Client> Get(int id)
        {
            return await Task.FromResult(_db.Clients.Find(id));
        }

        public async Task<IEnumerable<Client>> Find(Func<Client, bool> predicate)
        {
            return _db.Clients.Where(predicate).ToList();
        }

        public async Task Create(Client client)
        {
            _db.Clients.Add(client);
        }

        public async Task Update(Client client)
        {
            _db.Entry(client).State = EntityState.Modified;
        }
        public bool Delete(int id)
        {
            var client = _db.Clients.Find(id);
            if (client != null)
            {
                _db.Clients.Remove(client);
                return true;
            }

            return false;
        }
    }

public class HotelRepository : IRepository<Hotel>
    {
        private readonly HotelContext _db;

        public HotelRepository(HotelContext db)
        {
            _db = db;
        }

        public async Task<IEnumerable<Hotel>> GetAll()
        {
            return _db.Hotels;
        }

        public async Task<Hotel> Get(int id)
```

```
        {
            return _db.Hotels.Find(id);
        }

        public async Task<IEnumerable<Hotel>> Find(Func<Hotel, bool> predicate)
        {
            return _db.Hotels.Where(predicate).ToList();
        }

        public async Task Create(Hotel hotel)
        {
            _db.Hotels.Add(hotel);
        }

        public async Task Update(Hotel hotel)
        {
            _db.Entry(hotel).State = EntityState.Modified;
        }

        public bool Delete(int id)
        {
            var hotel = _db.Hotels.Find(id);
            if (hotel != null)
            {
                _db.Hotels.Remove(hotel);
                return true;
            }

            return false;
        }
    }

public class NutritionTypeRepository : IRepository<NutritionType>
    {
        private readonly HotelContext _db;

        public NutritionTypeRepository(HotelContext db)
        {
            _db = db;
        }
        public async Task<IEnumerable<NutritionType>> GetAll()
        {
            return _db.NutritionTypes;
        }

        public async Task<NutritionType> Get(int id)
        {
            return _db.NutritionTypes.Find(id);
        }

        public async Task<IEnumerable<NutritionType>> Find(Func<NutritionType, bool>
predicate)
        {
            return _db.NutritionTypes.Where(predicate).ToList();
        }

        public async Task Create(NutritionType nutritionType)
        {
            _db.NutritionTypes.Add(nutritionType);
        }

        public async Task Update(NutritionType nutritionType)
        {
            _db.Entry(nutritionType).State = EntityState.Modified;
        }
```

```
        public bool Delete(int id)
        {
            var nutritionType = _db.NutritionTypes.Find(id);
            if (nutritionType != null)
            {
                _db.NutritionTypes.Remove(nutritionType);
                return true;
            }

            return false;
        }
    }

public class RatingCriteriaRepository : IRepository<RatingCriteria>
    {
        private readonly HotelContext _db;

        public RatingCriteriaRepository(HotelContext db)
        {
            _db = db;
        }

        public async Task<IEnumerable<RatingCriteria>> GetAll()
        {
            return _db.RatingCriterias;
        }

        public async Task<RatingCriteria> Get(int id)
        {
            return await Task.FromResult(_db.RatingCriterias.Find(id));
        }

        public async Task<IEnumerable<RatingCriteria>> Find(Func<RatingCriteria, bool>
predicate)
        {
            return _db.RatingCriterias.Where(predicate).ToList();
        }

        public async Task Create(RatingCriteria ratingCriteria)
        {
            _db.RatingCriterias.Add(ratingCriteria);
        }

        public async Task Update(RatingCriteria ratingCriteria)
        {
            _db.Entry(ratingCriteria).State = EntityState.Modified;
        }
        public bool Delete(int id)
        {
            var ratingCriteria = _db.RatingCriterias.Find(id);
            if (ratingCriteria != null)
            {
                _db.RatingCriterias.Remove(ratingCriteria);
                return true;
            }

            return false;
        }
    }

public class ReservationRepository : IReservationRepository
    {
        private readonly HotelContext _db;
```

```
        public ReservationRepository(HotelContext db)
        {
            _db = db;
        }

        public async Task<IEnumerable<Reservation>> GetAll()
        {
            return _db.Reservations;
        }

        public async Task<Client> GetClientByReservation(int reservationId)
        {
            return _db.Reservations.Include(r => r.Client).First(r => r.Id ==
reservationId).Client;
        }
        public async Task<IEnumerable<Client>> GetClientsByHotel(int hotelId)
        {
            return _db.Reservations.Include(r => r.Client).Include(r => r.Room).Where(r =>
r.Room.HotelId == hotelId).Select(r => r.Client);
        }

        public async Task<Room> GetRoomByReservation(int reservationId)
        {
            return _db.Reservations.Include(r => r.Room).First(r => r.Id ==
reservationId).Room;
        }

        public async Task<Reservation> Get(int id)
        {
            return _db.Reservations.Find(id);
        }

        public async Task<IEnumerable<Reservation>> Find(Func<Reservation, bool> predicate)
        {
            return _db.Reservations.Where(predicate).ToList();
        }

        public async Task Create(Reservation reservation)
        {
            _db.Reservations.Add(reservation);
        }

        public async Task Update(Reservation reservation)
        {
            _db.Entry(reservation).State = EntityState.Modified;
        }

        public bool Delete(int id)
        {
            var reservation = _db.Reservations.Find(id);
            if (reservation != null)
            {
                _db.Reservations.Remove(reservation);
                return true;
            }

            return false;
        }
    }

public class RoomRepository : IRepository<Room>
    {
        private readonly HotelContext _db;

        public RoomRepository(HotelContext db)
```

```
        {
            _db = db;
        }

        public async Task<IEnumerable<Room>> GetAll()
        {
            return _db.Rooms;
        }

        public async Task<Room> Get(int id)
        {
            return _db.Rooms.Find(id);
        }

        public async Task Create(Room room)
        {
            _db.Rooms.Add(room);
        }

        public async Task Update(Room room)
        {
            _db.Entry(room).State = EntityState.Modified;
        }

        public async Task<IEnumerable<Room>> Find(Func<Room, bool> predicate)
        {
            return _db.Rooms.Where(predicate).ToList();
        }

        public bool Delete(int id)
        {
            var room = _db.Rooms.Find(id);

            if (room != null)
            {
                _db.Rooms.Remove(room);
                return true;
            }

            return false;
        }
    }

public class RoomTypeRepository : IRepository<RoomType>
    {
        private readonly HotelContext _db;

        public RoomTypeRepository(HotelContext db)
        {
            _db = db;
        }
        public async Task<IEnumerable<RoomType>> GetAll()
        {
            return _db.RoomTypes;
        }

        public async Task<RoomType> Get(int id)
        {
            return _db.RoomTypes.Find(id);
        }

        public async Task<IEnumerable<RoomType>> Find(Func<RoomType, bool> predicate)
        {
            return _db.RoomTypes.Where(predicate).ToList();
        }
```

```
        public async Task Create(RoomType roomType)
        {
            _db.RoomTypes.Add(roomType);
        }

        public async Task Update(RoomType roomType)
        {
            _db.Entry(roomType).State = EntityState.Modified;
        }

        public bool Delete(int id)
        {
            var roomType = _db.RoomTypes.Find(id);
            if (roomType != null)
            {
                _db.RoomTypes.Remove(roomType);
                return true;
            }

            return false;
        }
    }

public class UnitOfWork : IUnitOfWork
    {
        private readonly HotelContext _db;
        private ClientRepository _clientRepository;
        private HotelRepository _hotelRepository;
        private RoomRepository _roomRepository;
        private RoomTypeRepository _roomTypeRepository;
        private ReservationRepository _reservationRepository;
        private RatingCriteriaRepository _ratingCriteriaRepository;
        private NutritionTypeRepository _nutritionTypeRepository;
        private bool _disposed;

        public UnitOfWork()
        {
            _db = new HotelContext();
        }
        public IRepository<Hotel> Hotels => _hotelRepository ?? (_hotelRepository = new
HotelRepository(_db));

        public IRepository<Room> Rooms => _roomRepository ?? (_roomRepository = new
RoomRepository(_db));

        public IRepository<RoomType> RoomTypes => _roomTypeRepository ??
(_roomTypeRepository = new RoomTypeRepository(_db));
        public IRepository<NutritionType> NutritionTypes => _nutritionTypeRepository ??
(_nutritionTypeRepository = new NutritionTypeRepository(_db));
        public IRepository<Client> Clients => _clientRepository ?? (_clientRepository = new
ClientRepository(_db));
        public IRepository<RatingCriteria> RatingCriterias => _ratingCriteriaRepository ??
(_ratingCriteriaRepository = new RatingCriteriaRepository(_db));

        public IReservationRepository Reservations => _reservationRepository ??
(_reservationRepository = new ReservationRepository(_db));
        public void Save()
        {
            _db.SaveChanges();
        }
        protected virtual void Dispose(bool disposing)
        {
            if (!_disposed)
            {
```

```
                if (disposing)
                {
                    _db.Dispose();
                }
                _disposed = true;
            }
        }
        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }
    }
```

4 ІНТЕРФЕЙСИ

```
public interface IRepository<T> where T : class
    {
        Task<IEnumerable<T>> GetAll();

        Task<T> Get(int id);

        Task<IEnumerable<T>> Find(Func<T, bool> predicate);

        Task Create(T item);

        Task Update(T item);

        bool Delete(int id);
    }

public interface IReservationRepository
    {
        Task<IEnumerable<Reservation>> GetAll();
        Task<Client> GetClientByReservation(int reservationId);
        Task<Room> GetRoomByReservation(int reservationId);
        Task<IEnumerable<Client>> GetClientsByHotel(int hotelId);
        Task<Reservation> Get(int id);
        Task<IEnumerable<Reservation>> Find(Func<Reservation, bool> predicate);
        Task Create(Reservation item);
        Task Update(Reservation item);
        bool Delete(int id);
    }

public interface IUnitOfWork : IDisposable
    {
        IRepository<Room> Rooms { get; }
        IRepository<Hotel> Hotels { get; }
        IRepository<RoomType> RoomTypes { get; }
        IRepository<NutritionType> NutritionTypes { get; }
        IRepository<Client> Clients { get; }
        IRepository<RatingCriteria> RatingCriterias { get; }
        IReservationRepository Reservations { get; }
        void Save();
    }

public interface IClientService : IDisposable
    {
        Task<ClientDTO> GetClientById(int clientId);
        Task<IEnumerable<ClientDTO>> GetAllClients();
        Task AddClient(ClientDTO client);
```

```
        Task UpdateClient(ClientDTO client);
        bool DeleteClient(int clientId);
        Task<ClientDTO> GetClientByName(string clientFirstName, string clientLastName);
        Task<IEnumerable<ClientDTO>> GetClientsByRating(double rating);
        void SaveChanges();

    }

public interface IHotelService : IDisposable
    {
        Task<HotelDTO> GetHotelById(int hotelId);
        Task<IEnumerable<HotelDTO>> GetAllHotels();
        Task AddHotel(HotelDTO hotel);
        Task UpdateHotel(HotelDTO hotel);
        bool DeleteHotel(int hotelId);
        Task<HotelDTO> GetHotelByName(string hotelName);
        Task<IEnumerable<HotelDTO>> GetHotelsByCity(string city);
        Task<IEnumerable<HotelDTO>> GetHotelsByRating(int rating);
        Task<IEnumerable<HotelDTO>> GetHotelsByNutrition(int nutritionTypeId);
        Task<IEnumerable<HotelDTO>> GetHotelsByRoomCleaning(bool hasRoomCleaning);
        Task<IEnumerable<HotelDTO>> GetHotelsByParking(bool hasParking);
        Task<IEnumerable<HotelDTO>> GetHotelsByRatingArray(int rat1, int rat2, int rat3);
        Task<IEnumerable<HotelDTO>> GetHotelsByDistanceArray(double dist1, double dist2,
double dist3);
        Task<IEnumerable<HotelDTO>> GetHotelsByNutritionArray(int ro, int bb, int hb, int
fb, int ai);
        void SaveChanges();
    }

public interface INutritionTypeService : IDisposable
    {
        Task<NutritionTypeDTO> GetNutritionTypeById(int nutritionTypeId);
        Task<IEnumerable<NutritionTypeDTO>> GetAllNutritionTypes();
        Task AddNutritionType(NutritionTypeDTO nutritionType);
        Task UpdateNutritionType(NutritionTypeDTO nutritionType);
        bool DeleteNutritionType(int nutritionTypeId);
        void SaveChanges();
    }

public interface IRatingCriteriaService : IDisposable
    {
        Task<RatingCriteriaDTO> GetRatingById(int ratingCriteriaId);
        Task<IEnumerable<RatingCriteriaDTO>> GetAllRatings();
        Task AddRating(RatingCriteriaDTO ratingCriteria);
        Task UpdateRating(RatingCriteriaDTO ratingCriteria);
        bool DeleteRating(int ratingid);
        Task<IEnumerable<RatingCriteriaDTO>> GetRatingByClientId(int clientId);
        Task<IEnumerable<RatingCriteriaDTO>> GetRatingByByHotelId(int hotelId);
        void SaveChanges();
    }

public interface IReservationService : IDisposable
    {
        Task<ReservationDTO> GetReservationById(int reservationId);
        Task<IEnumerable<ReservationDTO>> GetAllReservations();
        Task AddReservation(ReservationDTO reservation);
        Task UpdateReservation(ReservationDTO reservation);
        bool DeleteReservation(int reservationId);
        Task<IEnumerable<ReservationDTO>> GetReservationsByDate(DateTime bookingDate,
DateTime bookingDateEnd);
        Task<ClientDTO> GetClientByReservation(int reservationId);
        Task<IEnumerable<ClientDTO>> GetClientsByHotel(int hotelId);
        Task<RoomDTO> GetRoomByReservation(int reservationId);
        void SaveChanges();
    }
```

```
public interface IRoomService : IDisposable
    {
        Task<RoomDTO> GetRoomById(int roomId);
        Task AddRoom(RoomDTO room);
        Task UpdateRoom(RoomDTO room);
        bool DeleteRoom(int roomId);
        Task<IEnumerable<RoomDTO>> GetRoomsByRoomType(int roomTypeId);
        Task<IEnumerable<RoomDTO>> GetRoomsByHotel(int hotelId);
        void SaveChanges();
    }

public interface IRoomTypeService : IDisposable
    {
        Task<RoomTypeDTO> GetRoomTypeById(int roomTypeId);
        Task<IEnumerable<RoomTypeDTO>> GetAllRoomTypes();
        //Task<IEnumerable<RoomTypeDTO>> GetRoomTypesByPrice(double minPrice, double
maxPrice);
        Task AddRoomType(RoomTypeDTO roomType);
        Task UpdateRoomType(RoomTypeDTO roomType);
        bool DeleteRoomType(int roomTypeId);
        void SaveChanges();
    }
```

5 СУТНОСТІ

```
public class Client
    {
        public int Id { get; set; }
        [MaxLength(100)]
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public double Rating { get; set; }
        public IEnumerable<Reservation> Reservations { get; set; }
    }

public class Hotel
    {

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        [Key, Column(Order = 0)]
        public int Id { get; set; }
        [MaxLength(50)]
        public string Name { get; set; }
        [MaxLength(50)]
        public int NutritionTypeId { get; set; }
        public NutritionType NutritionType { get; set; }
        public string City { get; set; }
        public string Address { get; set; }
        public string Description { get; set; }
        public bool HasRoomCleaning { get; set; }
        public bool HasParking { get; set; }
        public double DistanceToCityCenter { get; set; }
        public int Rating { get; set; }
    }

public class NutritionType
    {
        public int Id { get; set; }
        [MaxLength(25)]
        public string Name { get; set; }
        [MaxLength(200)]
        public string Description { get; set; }
        public bool IsPaid { get; set; }
```

```
    }

public class RatingCriteria
    {
        public int Id { get; set; }
        public int ClientId { get; set; }
        public Client Client { get; set; }
        public int HotelId { get; set; }
        public Hotel Hotel { get; set; }
        [Range(1, 10, ErrorMessage = "Value must be between 1 to 10")]
        public int Decency { get; set; }
        [Range(1, 10, ErrorMessage = "Value must be between 1 to 10")]
        public int RoomCleanliness { get; set; }
        [Range(1, 10, ErrorMessage = "Value must be between 1 to 10")]
        public int HotelRulesCompliance { get; set; }
        [Range(1, 10, ErrorMessage = "Value must be between 1 to 10")]
        public int Behavior { get; set; }
        [MaxLength(100)]
        public string Comment { get; set; }
    }

public class Reservation
    {
        public int Id { get; set; }
        public int ClientId { get; set; }
        public Client Client { get; set; }
        public int RoomId { get; set; }
        public Room Room { get; set; }
        public DateTime BookingDate { get; set; }
        public DateTime BookingDateEnd { get; set; }

    }

public class Room
    {
        public int Id { get; set; }
        public int RoomTypeId { get; set; }
        public RoomType RoomType { get; set; }
        public int HotelId { get; set; }
        public Hotel Hotel { get; set; }
        public bool IsReserved { get; set; }
        public double Price { get; set; }
        public IEnumerable<Reservation> Reservations { get; set; }
    }

public class RoomType
    {
        public int Id { get; set; }
        [MaxLength(25)]
        public string Name { get; set; }
        [MaxLength(200)]
        public string Description { get; set; }
        //public double Price { get; set; }
        public int Capacity { get; set; }
    }
```

6 VUE КОМПОНЕНТИ

Login.vue

```
<template>
  <div class="bg-image">
    <div class="login-basic1">
      <div class="card-body-header1" style="font-
family: CURSIVE">Zaiets Booking</div>
      <div style="padding: 10px;">
        <form>
          <p
            class="h4 text-center mb-4 padding-title"
            style="color:#2bbbad;font-weight: bold;font-size: 40px;font-
family: CURSIVE"
          >Sign in</p>
          <div class="grey-text">
            <mdb-input label="Your email" icon="envelope" type="email" v-
model="email" />
            <br />
            <mdb-input label="Your password" icon="lock" type="password" v-
model="password" />
            <div style="display:flex; justify-content:space-between;">
              <div style="color:#757575;margin-bottom:15px">
              <input type="checkbox"> Remember me
            </div>
            <div style="cursor:pointer;color:#2bbbad;font-
weight:bold;" @click="goToRegister">
              Create account
            </div>
            </div>
          </div>
          <div class="text-center">
            <mdb-btn
              style="width:150px;padding:10px;font-
family: CURSIVE;color: white;background-color: #2bbbad !important;"
              @click="authorise();goToHome()"
            >Login</mdb-btn>
          </div>
        </form>
      </div>
    </div>
  </div>
</template>


<script>
import { mdbInput, mdbBtn } from "mdbvue";
import Vue from "vue";
import axios from "axios";
// import Loader from "../components/Loader";
```

```
export default Vue.extend({
  name: "Basic",
  components: {
    mdbInput,
    mdbBtn
    // Loader
  },
  data() {
    return {
      email: null,
      password: null,
      url:null
    };
  },
  methods: {
    authorise() {
      let email = this.email;
      localStorage.setItem("userEmail", email)
      let password = this.password;
      let url = this.url;
      axios
        .post("https://localhost:5001/api/account/login", {
          email,
          password,
          url
        })
        .then(function(response) {
          setTimeout(() => {
            console.log(response);
          }, 100);
        })
        .catch(function(error) {
          console.log(error);
          alert("Login Error!");
        });
    },
    goToHome() {
      if(this.email == null || this.password == 0){
        alert("Wrong data!");
        return;
      }
      this.$router.push("/");
    },
    goToRegister(){
      this.$router.push("/registration");
    }
  }
});
</script>
```

```
<style>
.login-basic1 {
  left: 42%;
  width: 350px;
  position: relative;
  top: 200px;
  border: 3px solid #2bbbad;
  border-radius: 8px;
  background-color: #e4f2ef;
}
.card-body-header1 {
  height: 90px;
  background-color: #2bbbad;
  color: #ffffff;
  align-items: center;
  text-align: center;
  display: flex;
  padding-left: 2.5rem;
  font-size: 1.5rem;
  font-weight: bold;
  color: aliceblue;
  padding-left: 90px;
}
.padding-title {
  padding-top: 25px;
  padding-bottom: 25px;
}
.bg-image {
  background-image: url(../../public/images/loginBG.jpg);
  background-repeat: no-repeat;
  background-size: cover;
  background-position: center;
  height: 100vh;
  width: 100vw;
}
</style>
```

Reservation.vue

```
<template>
  <div style="background-color:#99d6ff;min-height:100vh;">
    <div>
      <NavBar />
      <div class="login-basic">
        <div
          class="card-body-header"
          style="font-family: CURSIVE; justify-content:center;padding-left:0"
        >Please, fill all nececary fields</div>
        <div style="padding: 10px;">
          <form>
```

```
            <div style="display: flex; padding-left:200px">
              <p
                class="h4 text-center mb-4 padding-title"
                style="color:#3D99F5;font-weight: bold;font-size: 40px;font-
family: CURSIVE"
              >Reservation</p>
              <div
                style="padding-top:35px;position: absolute;right: 100px;font-
size: 22px;"
              >status: New</div>
            </div>
            <div class="grey-text">
              <mdb-input
                style="padding-bottom:10px"
                label="Email"
                icon="envelope"
                type="email"
                v-model="email"
              />
              <mdb-input
                style="padding-bottom:10px"
                label="Confirm Email"
                icon="exclamation-triangle"
                type="email"
                v-model="name"
              />
              <div style="display:flex;">
                <div style="margin-top:25px">
                  <vue-
dropdown :config="config" @setSelectedOption="setNewSelectedOption($event)"></vue-
dropdown>
                </div>
                <div style="mirgin-top:0px !important; padding-left:30px;">
                  <mdb-input label="First name" icon="user" type="text" v-
model="name" />
                </div>
                <div style="mirgin-top:0px !important; padding-left:30px;">
                  <mdb-input label="Last name" icon="user" type="text" v-
model="name" />
                </div>
              </div>
            </div>
            <div class="text-center">
              <mdb-btn
                style="width:150px;padding:10px;font-family: CURSIVE;"
                class="btn-default"
                @click="reserve()"
              >Reserve</mdb-btn>
            </div>
          </form>
        </div>
```

```
      </div>
      <!-- <Footer style="bottom:0" />  -->
    </div>
  </div>
</template>

<script>
import Vue from "vue";
// import axios from "axios";
import NavBar from "./NavBar";
// import Footer from "./Footer";
import { mdbInput, mdbBtn } from "mdbvue";
import VueDropdown from "vue-dynamic-dropdown";

export default Vue.extend({
  components: {
    NavBar,
    // Footer,
    mdbInput,
    mdbBtn,
    VueDropdown
  },
  data() {
    return {
      config: {
        options: [
          {
            value: "Mr."
          },
          {
            value: "Ms."
          },
          {
            value: "Mrs."
          }
        ],
        placeholder: "Title",
        backgroundColor: "#97CEF7",
        textColor: "black",
        borderRadius: "1.1em",
        border: "1px solid #3D99F5",
        width: 180
      }
    };
  },
  methods: {
    reserve() {},
    setNewSelectedOption(selectedOption) {
      this.config.placeholder = selectedOption.value;
    }
  }
```

```
});
</script>

<style>
.login-basic {
  left: 32%;
  width: 700px;
  position: relative;
  top: 120px;
  border: 3px solid #3d99f5;
  border-radius: 8px;
  background-color: #e4f2ef;
}
.card-body-header {
  height: 90px;
  background-color: #3d99f5;
  color: #ffffff;
  align-items: center;
  text-align: center;
  display: flex;
  padding-left: 2.5rem;
  font-size: 1.5rem;
  font-weight: bold;
  color: aliceblue;
  padding-left: 90px;
}
.padding-title {
  padding-top: 25px;
  padding-bottom: 25px;
}
.btn-default {
  color: white;
  background-color: #3d99f5 !important;
}
</style>
```

Register.vue

```
<template>
  <div class="bg-image">
    <div class="login-basic2">
      <div class="card-body-header2" style="font-
family: CURSIVE">Zaiets Booking</div>
      <div style="padding: 10px;">
        <form>
          <p
            class="h4 text-center mb-4 padding-title"
            style="color:#2bbbad;font-weight: bold;font-size: 40px;font-
family: CURSIVE"
          >Create account</p>
```

```
          <div class="grey-text">
            <mdb-input
              style="padding-bottom:10px"
              label="Email"
              icon="envelope"
              type="email"
              v-model="email"
            />
            <mdb-input
              style="padding-bottom:10px"
              label="Username"
              icon="user"
              type="text"
              v-model="name"
            />
            <mdb-input
              style="padding-bottom:10px"
              label="Password"
              icon="lock"
              type="password"
              v-model="password"
            />
            <mdb-input
              style="padding-bottom:10px"
              label="Confirm Password"
              icon="exclamation-triangle"
              type="password"
              v-model="passwordConfirm"
            />
          </div>
          <div class="text-center">
            <mdb-btn
              style="width:150px;padding:10px;font-
family: CURSIVE;color: white;background-color: #2bbbad !important;"
              @click="register()"
            >Create</mdb-btn>
            <mdb-btn
              style="width:150px;padding:10px;font-
family: CURSIVE;color: white;background-color: #2bbbad !important;"
              @click="varifyUser()"
            >Sign In</mdb-btn>
          </div>
        </form>
      </div>
    </div>
  </div>
</template>


<script>
import { mdbInput, mdbBtn } from "mdbvue";
```

```
import Vue from "vue";
import axios from "axios";

export default Vue.extend({
  name: "Basic",
  components: {
    mdbInput,
    mdbBtn
  },
  data() {
    return {
      email: null,
      name: null,
      password: null,
      passwordConfirm: null,
      isExist: false
    };
  },
  methods: {
    register() {
      let email = this.email;
      let name = this.name;
      localStorage.setItem("userName", name);
      let password = this.password;
      let passwordConfirm = this.passwordConfirm;
      axios
        .post("https://localhost:5001/api/account/register", {
          email,
          name,
          password,
          passwordConfirm
        })
        .then(function(response) {
          console.log(response);
          alert("Success!");
        })
        .catch(function(error) {
          console.log(error);
          alert("Validation error!");
        });
    },
    varifyUser() {
      let user = this.user;
      axios
        .get(
          "https://localhost:5001/api/account/register/confirm/" + this.email
        )
        .then(response => {
          setTimeout(() => {
            user = response.data;
            if (user.email == this.email) {
```

```
              this.isExist = true;
              this.goToLogin();
            }
          }, 100);
        })
        .catch(error => {
          console.log(error);
        });
      this.goToLogin();
    },
    goToLogin() {
      if (this.isExist == true) {
        this.$router.push("/login");
      }
    }
  }
});
</script>

<style>
.login-basic2 {
  left: 42%;
  width: 350px;
  position: relative;
  top: 120px;
  border: 3px solid #2bbbad;
  border-radius: 8px;
  background-color: #e4f2ef;
}
.card-body-header2 {
  height: 90px;
  background-color: #2bbbad;
  color: #ffffff;
  align-items: center;
  text-align: center;
  display: flex;
  padding-left: 2.5rem;
  font-size: 1.5rem;
  font-weight: bold;
  color: aliceblue;
  padding-left: 90px;
}
.padding-title {
  padding-top: 25px;
  padding-bottom: 25px;
}
.bg-image {
  background-image: url(../../public/images/loginBG.jpg);
  background-repeat: no-repeat;
  background-size: cover;
  background-position: center;
```

```
  height: 100vh;
  width: 100vw;
}
</style>
```

Home.vue

```
<template>
  <div style="background-color:#99d6ff;min-height:100vh">
    <NavBar />
    <div v-if="collapsed" class="banner">
      <div>
        <i class="mdi mdi-information-outline info-
button" style="color: #FF0000"></i>
        Coronavirus(COVID-19): info
      </div>
      <div class="info-eye">
        <i class="mdi mdi-eye" @click="collapsed = !collapsed"></i>
      </div>
    </div>
    <div v-else class="banner-expanded">
      <div>
        <i class="mdi mdi-information-outline info-
button" style="color: #FF0000"></i>
        Coronavirus(COVID-19): info
        <p>
          Booking unavailable during pandemic.
          Please, stay at home. Together we will stop it!
        </p>
      </div>
      <div class="info-eye">
        <i class="mdi mdi-eye" @click="collapsed = !collapsed"></i>
      </div>
    </div>
    <p class="calendar-title">Choose date for your trip</p>
    <div style="display:flex;justify-content:center;">
      <div style="margin-top:30px">
        <b-row>
          <b-col md="auto" class="no-padding">
            <b-calendar
              v-model="value1"
              @context="onContextFrom"
              locale="en-US"
              style="border: 2px solid #3d99f5;"
            ></b-calendar>
          </b-col>
        </b-row>
      </div>
      <div style="margin-top:30px; padding-left:100px">
        <b-row>
```

```
          <b-col md="auto" class="no-padding">
            <b-calendar
              v-model="value2"
              @context="onContextTo"
              locale="en-US"
              style="border: 2px solid #3d99f5;"
            ></b-calendar>
          </b-col>
        </b-row>
      </div>
    </div>
    <div class="calendar">
      <div>
        <b>From:</b>
        {{contextFrom}}
        <i
          class="mdi mdi-arrow-right-bold-outline"
          style="padding: 0 10px 0 10px;font-size:x-large;"
        ></i>
        <b>To:</b>
        {{contextTo}}
      </div>
    </div>
    <b-button href="/hotels" variant="primary" class="btn-hotels">To hotels</b-
button>
    <div class="cities-title">The most popular cities</div>
    <div
      style="display:flex; margin-top: 30px; justify-content:center;padding-
bottom:50px;font-family:cursive;"
    >
      <b-card
        title="Chernigiv"
        img-src="../../public/images/che.jpg"
        img-alt="Image"
        img-top
        img-height="212px"
        tag="article"
        style="max-width: 20rem; margin-right:30px;cursor:pointer"
        class="mb-2 card-image-top"
      >
        <b-card-
text>Beautiful city with lots of ancient architecture and places for entertainment.
 You have passibility to...</b-card-text>
      </b-card>
      <b-card
        title="Kyiv"
        img-src="../../public/images/kyiv.jpg"
        img-alt="Image"
        img-top
        img-height="212px"
        tag="article"
```

```
        style="max-width: 20rem; margin-right:30px;cursor:pointer"
        class="mb-2 card-image-top"
      >
        <b-card-
text>Beautiful city with lots of ancient architecture and places for entertainment.
 You have passibility to...</b-card-text>
      </b-card>
      <b-card
        title="Kharkiv"
        img-src="../../public/images/kharkiv.jpg"
        img-alt="Image"
        img-top
        img-height="212px"
        tag="article"
        style="max-width: 20rem;  margin-right:30px;cursor:pointer"
        class="mb-2 card-image-top"
      >
        <b-card-
text>Beautiful city with lots of ancient architecture and places for entertainment.
 You have passibility to...</b-card-text>
      </b-card>
      <b-card
        title="Lviv"
        img-src="../../public/images/lviv.jpg"
        img-alt="Image"
        img-top
        img-height="212px"
        tag="article"
        style="max-width: 20rem; margin-right:30px;cursor:pointer"
        class="mb-2 card-image-top"
      >
        <b-card-
text>Beautiful city with lots of ancient architecture and places for entertainment.
 You have passibility to...</b-card-text>
      </b-card>
    </div>
    <Footer />
  </div>
</template>

<script>
import Vue from "vue";
import NavBar from "./NavBar";
import Footer from "./Footer"

export default Vue.extend({
  data() {
    return {
      value1: "",
      value2: "",
      contextFrom: null,
```

```
      contextTo: null,
      collapsed: true
    };
  },
  components: {
    NavBar,
    Footer
  },
  methods: {
    onContextFrom(ctx) {
      this.contextFrom = ctx.selectedFormatted;
    },
    onContextTo(ctx1) {
      this.contextTo = ctx1.selectedFormatted;
    }
  }
});
</script>

<style>
.banner {
  width: 100%;
  height: 50px;
  background-color: #fff8e1;
  padding-left: 100px;
  align-items: center;
  display: flex;
  font-weight: bold;
  font-family: cursive;
  justify-content: space-between;
}
.banner-expanded {
  width: 100%;
  height: 100px;
  background-color: #fff8e1;
  padding-left: 100px;
  align-items: center;
  display: flex;
  font-weight: bold;
  font-family: cursive;
  justify-content: space-between;
}
.info-button {
  padding-right: 10px;
  font-size: x-large;
}
.info-eye {
  padding-right: 50px;
  cursor: pointer;
  font-size: x-large;
}
```

```
.no-padding {
  padding-left: 0px !important;
  padding-right: 0px !important;
}
.calendar {
  border: 2px solid #3d99f5;
  background-color: white;
  width: 25%;
  height: 50px;
  display: flex;
  justify-content: center;
  align-items: center;
  left: 34%;
  margin-top: 20px;
  position: relative;
}
.calendar-title {
  position: relative;
  justify-content: center;
  display: flex;
  margin-top: 25px;
  font-family: cursive;
  color: black;
  font-weight: bold;
  font-size: 30px;
}
.cities-title {
  position: relative;
  display: flex;
  margin-top: 25px;
  font-family: cursive;
  color: black;
  font-weight: bold;
  font-size: 30px;
  padding-left: 13%;
}
.btn-hotels {
  position: relative;
  left: 60%;
  top: -51px;
}

</style>
```

Account.vue

```
<template>
  <div style="background-color:#99d6ff;min-height:100vh;font-family:cursive">
    <NavBar />
    <div class="login-basic">
```

```
        <div
          class="card-body-header"
          style="font-family: CURSIVE; justify-content:center;"
        >
            <b-avatar style="padding-right:12px;right:35px" size="72px"></b-avatar>
        </div>
        <div style="padding: 10px;">
          <form>
            <div style="display: flex; padding-left:150px">
              <p
                class="h4 text-center mb-4 padding-title"
                style="color:#3D99F5;font-weight: bold;font-size: 40px;font-
family: CURSIVE;"
              >Your personal account</p>
            </div>
            <div class="grey-text">
                <b-form-file
                    v-model="file"
                    :state="Boolean(file)"
                    placeholder="Choose your avatar"
                    drop-placeholder="Drop file here..."
                ></b-form-file>
                <div class="mt-
3">Selected file: {{ file ? file.name : '' }}</div>
              <mdb-input label="Old password" icon="lock" type="password" v-
model="password" />
              <mdb-input label="New password" icon="lock" type="password" v-
model="newPassword" />
              <div style="display:flex;">
                <div style="justify-
content:center; display:flex; position:relative;left:80px">
                    <div style="mirgin-top:0px !important; padding-left:30px;">
                  <mdb-input label="First name" icon="user" type="text" v-
model="name" />
                </div>
                <div style="mirgin-top:0px !important; padding-left:30px;">
                  <mdb-input label="Last name" icon="user" type="text" v-
model="name" />
                </div>
                    </div>

              </div>
            </div>
            <div class="text-center">
              <mdb-btn
                style="width:150px;padding:10px;font-family: CURSIVE;"
                class="btn-default"
                @click="Update()"
              >Update</mdb-btn>
            </div>
          </form>
```

```
        </div>
      </div>
    <Footer style="bottom:0"/>
  </div>
</template>

<script>
import Vue from "vue";
import NavBar from "./NavBar";
import Footer from "./Footer";
import { mdbInput, mdbBtn } from "mdbvue";

export default Vue.extend({
    components:{
        NavBar,
        Footer,
        mdbInput,
        mdbBtn,
    },
    data(){
        return{
            file: null,
            password: null,
            newPassword: null
        }
    },
    methods:{
        Update(){
             this.$router.push("/");
        }
    }
})
</script>

<style>
.login-basic {
  left: 32%;
  width: 750px;
  position: relative;
  top: 50px;
  background-color: #e4f2ef;
}
</style>
```

Rooms.vue

```
<template>
  <div style="background-color:#99d6ff;min-height:100vh">
    <div>
      <NavBar />
```

```
      <div>
        <div class="title-rooms">
          <h1 style="font-size:3rem !important">Rooms</h1>
        </div>

        <Loader v-if="loading" />
        <div
          v-else-if="rooms.length"
          style="display:flex; margin-top: 30px; justify-content:center;padding-
bottom:50px;font-family:cursive;"
        >
          <b-card
            title="Suite"
            img-src="../../../public/images/suite.jpg"
            img-alt="Image"
            img-top
            img-height="212px"
            tag="article"
            style="max-width: 20rem; margin-right:30px;cursor:pointer"
            class="mb-2 card-image-top"
            :class="{ 'disable-form' : roomTypeCounter[0].count == 0}"
          >
            <b-card-
text>A room comprises of two or more bedroom, a living room and a dining area. Belo
ng to the luxury category. Demanded by tourists, especially families.</b-card-text>
            <div class="room-card">
              <b-
button squared variant="primary" @click="goToReserve()">Reserve</b-button>
              <div>Price: ${{roomTypeCounter[0].price}}</div>
            </div>
            <div class="available">Available:{{roomTypeCounter[0].count}}</div>
          </b-card>
          <b-card
            title="Standart"
            img-src="../../../public/images/standart.jpg"
            img-alt="Image"
            img-top
            img-height="212px"
            tag="article"
            style="max-width: 20rem; margin-right:30px;cursor:pointer"
            class="mb-2 card-image-top"
            :class="{ 'disable-form' : roomTypeCounter[1].count == 0}"
          >
            <b-card-
text>Standard single rooms with minimum comfort. Has double beb facility.There is a
 TV, telephone, refrigerator, minimal furniture, a bathroom.</b-card-text>
            <div class="room-card">
              <b-
button squared variant="primary" @click="goToReserve()">Reserve</b-button>
              <div>Price: ${{roomTypeCounter[1].price}}</div>
            </div>
```

```
            <div class="available">Available:{{roomTypeCounter[1].count}}</div>
          </b-card>
          <b-card
            title="De Luxe"
            img-src="../../../public/images/de_luxe.jpg"
            img-alt="Image"
            img-top
            img-height="212px"
            tag="article"
            style="max-width: 20rem;  margin-right:30px;cursor:pointer"
            class="mb-2 card-image-top"
            :class="{ 'disable-form' : roomTypeCounter[2].count == 0}"
          >
            <b-card-
text>Superior rooms which mainly bigger than others. Have all necessary appliances
with excellent quality. The deluxe rooms have all the necessary appliances.</b-
card-text>
            <div class="room-card">
              <b-
button squared variant="primary" @click="goToReserve()">Reserve</b-button>
              <div>Price: ${{roomTypeCounter[2].price}}</div>
            </div>
            <div class="available">Available:{{roomTypeCounter[2].count}}</div>
          </b-card>
          <b-card
            title="Duplex"
            img-src="../../../public/images/duplex.jpg"
            img-alt="Image"
            img-top
            img-height="212px"
            tag="article"
            style="max-width: 20rem; margin-right:30px;cursor:pointer"
            class="mb-2 card-image-top"
            :class="{ 'disable-form' : roomTypeCounter[3].count == 0}"
          >
            <b-card-
text>A room which is been spread on two floars connected by an internal staircase.
Have all necessary appliances with excellent quality.</b-card-text>
            <div class="room-card">
              <b-
button squared variant="primary" @click="goToReserve()">Reserve</b-button>
              <div>Price: ${{roomTypeCounter[3].price}}</div>
            </div>
            <div class="available">Available:{{roomTypeCounter[3].count}}</div>
          </b-card>
        </div>
      </div>
    </div>
    <Footer style="bottom:0" />
  </div>
</template>
```

```
<script>
import Vue from "vue";
import axios from "axios";
import Loader from "../Loader";
import NavBar from "../NavBar";
import Footer from "../Footer";

export default Vue.extend({
  components: {
    Loader,
    NavBar,
    Footer
  },
  data() {
    return {
      hotelId: localStorage.getItem("hotelId"),
      rooms: [],
      roomTypeCounter: [
        { count: 0, price: 0 },
        { count: 0, price: 0 },
        { count: 0, price: 0 },
        { count: 0, price: 0 }
      ],
      loading: true
    };
  },
  methods: {
    goToReserve() {
      console.log("qweqwe");
      this.$router.push("/hotels/*/rooms/reservation");
    }
  },
  mounted() {
    axios
      .get("https://localhost:5001/api/rooms/hotel/" + this.hotelId)
      .then(response => {
        setTimeout(() => {
          this.rooms = response.data;
          this.rooms.forEach(element => {
            if (element.roomTypeId == 1) {
              this.roomTypeCounter[0].count++;
              this.roomTypeCounter[0].price = element.price;
            } else if (element.roomTypeId == 2) {
              this.roomTypeCounter[1].count++;
              this.roomTypeCounter[1].price = element.price;
            } else if (element.roomTypeId == 3) {
              this.roomTypeCounter[2].count++;
              this.roomTypeCounter[2].price = element.price;
            } else if (element.roomTypeId == 4) {
              this.roomTypeCounter[3].count++;
```

```
            this.roomTypeCounter[3].price = element.price;
          }
          this.loading = false;
        });
        console.log(this.roomTypeCounter);
      }, 600);
    })
    .catch(error => {
      console.log(error);
    });
  }
});
</script>

<style>
.title-rooms {
  text-align: center;
  margin: 50px auto 70px auto;
  font-weight: bold;
  font-family: cursive;
}
.room-card {
  display: flex;
  justify-content: space-between;
  align-items: center;
  font-weight: bold;
  font-size: 20px;
}
.disable-form {
  background-color: #e2dddd;
  opacity: 0.7;
  pointer-events: none;
}
.available {
  padding: 10px 10px 0px 10px;
  font-weight: bold;
  font-size: 18px;
}
</style>
```

Hotels.vue

```
<template>
  <div>
    <div style="background-color:#99d6ff;min-height:100vh;font-family:cursive">
      <NavBar />
      <div class="title-design">
        <h1 style="font-family:cursive">All hotels</h1>
      </div>
      <div class="filer">
```

```
        <b-card no-body style="min-width: 15rem;max-width:15rem">
          <template v-slot:header>
            <h4 style="text-align: center;font-weight:bold" class="mb-
0">Filter by:</h4>
          </template>
          <b-card-body>
            <b-button
              style="justify-content: center;font-
weight:bold;position: relative;left: 35px"
              squared
              variant="primary"
              @click="resetAll()"
            >Reset all</b-button>
            <b-card-text style="font-weight:bold;font-
size:20px;color:black">Star rating</b-card-text>
            <b-list-group>
              <b-form-checkbox v-
model="stars3" value="3" name="stars" id="s1">3 stars</b-form-checkbox>
              <b-form-checkbox v-
model="stars4" value="4" name="stars" id="s2">4 stars</b-form-checkbox>
              <b-form-checkbox v-
model="stars5" value="5" name="stars" id="s3">5 stars</b-form-checkbox>
            </b-list-group>
          </b-card-body>
          <b-card-body>
            <b-card-text style="font-weight:bold;font-
size:20px;color:black">Nutrition</b-card-text>
            <b-list-group>
              <b-form-checkbox v-
model="ro" value="1" name="nutrition" id="n1">RO(Room only)</b-form-checkbox>
              <b-form-checkbox v-
model="bb" value="2" name="nutrition" id="n2">BB(Bed & breakfast)</b-form-checkbox>
              <b-form-checkbox v-
model="hb" value="3" name="nutrition" id="n3">HB(Half board)</b-form-checkbox>
              <b-form-checkbox v-
model="fb" value="4" name="nutrition" id="n4">FB(Full board)</b-form-checkbox>
              <b-form-checkbox v-
model="ai" value="5" name="nutrition" id="n5">AI(All inclusive)</b-form-checkbox>
            </b-list-group>
          </b-card-body>
          <b-card-body>
            <b-card-text style="font-weight:bold;font-
size:20px;color:black">Has room cleaning</b-card-text>
            <b-list-group>
              <b-form-radio v-
model="hasRoomCleaning" name="radio1" value="true">Yes</b-form-radio>
              <b-form-radio v-
model="hasRoomCleaning" name="radio2" value="false">No</b-form-radio>
            </b-list-group>
          </b-card-body>
          <b-card-body>
```

```
            <b-card-text style="font-weight:bold;font-
size:20px;color:black">Has parking</b-card-text>
            <b-list-group>
              <b-form-radio v-model="hasParking" name="radio3" value="true">Yes</b-
form-radio>
              <b-form-radio v-model="hasParking" name="radio4" value="false">No</b-
form-radio>
            </b-list-group>
          </b-card-body>
          <b-card-body>
            <b-card-text style="font-weight:bold;font-size:20px;color:black">
              Distance to center
              <center></center>
            </b-card-text>
            <b-list-group>
              <b-form-checkbox v-
model="distance1" value="1" name="distance" id="d1">&#60; 1 km</b-form-checkbox>
              <b-form-checkbox v-
model="distance2" value="2" name="distance" id="d2">1-2 km</b-form-checkbox>
              <b-form-checkbox v-
model="distance3" value="3" name="distance" id="d3">&#62; 2 km</b-form-checkbox>
            </b-list-group>
          </b-card-body>
          <b-button squared variant="primary" @click="filter">Filtered</b-button>
        </b-card>
      </div>
      <Loader v-if="loading" />
      <div v-else-if="hotels.length">
        <div
          v-for="hotel in hotels"
          v-bind:key="hotel.id"
          class="div hotelCard"
          style="font-family:cursive"
        >
          <b-card no-body style="min-width: 25rem;max-width:25rem">
            <!-- Image carousel -->
            <div>
              <b-carousel
                id="carousel-1"
                :interval="500000"
                controls
                img-width="320"
                img-height="150"
                style="text-shadow: 1px 1px 2px #333;"
              >
                <b-carousel-slide
                  img-height="225px"
                  img-src="../../../public/images/hotel.jpg"
                  class="img-fluid"
                ></b-carousel-slide>
                <b-carousel-slide
```

```
                  img-height="225px"
                  img-src="../../../public/images/hotel2.jpg"
                  class="img-fluid"
                ></b-carousel-slide>
                <b-carousel-slide
                  img-height="225px"
                  img-src="../../../public/images/reception.png"
                  class="img-fluid"
                ></b-carousel-slide>
              </b-carousel>
            </div>

            <!-- Card body -->
            <template v-slot:header>
              <h4 style="text-align: center;font-weight:bold" class="mb-
0">{{hotel.name}}</h4>
            </template>
            <b-list-group-item class="rating-style more-
info" :style="ratingColor(hotel.rating)">
              Stars: {{hotel.rating}}
              <div>
                <b style="font-size:16px;color:black">More</b>
                <i
                  class="mdi mdi-arrow-bottom-left-thick"
                  style="color:#3d99f5;"
                  @click="detailsShow=!detailsShow"
                ></i>
              </div>
            </b-list-group-item>

            <div v-show="detailsShow">
              <b-card-body>
                <b-card-title>Description</b-card-title>
                <b-card-text>{{hotel.description}}</b-card-text>
              </b-card-body>
              <b-list-group>
                <b-list-group-item>City: {{hotel.city}}</b-list-group-item>
                <b-list-group-item>Address: {{hotel.address}}</b-list-group-item>
              </b-list-group>
              <b-card-body>
                <b-card-
title>Nutrition: {{getNutritionByHotelId(hotel.nutritionTypeId)}}</b-card-title>
                <b-card-text>teet a afdsf dsf</b-card-text>
              </b-card-body>
              <b-list-group>
                <b-list-group-item>Room cleaning: {{hotel.hasRoomCleaning}}</b-
list-group-item>
                <b-list-group-item>Parking: {{hotel.hasParking}}</b-list-group-
item>
```

```
              <b-list-group-
item>Distance to the city center: {{hotel.distanceToCityCenter}}</b-list-group-
item>
            </b-list-group>
          </div>

          <b-card-footer style="text-align:center;font-weight: bold; font-
size: 25px;">
            <a
              :href="goToRooms(hotel.id)"
              @click="setHotelId(hotel.id)"
              class="card-link"
            >Go to rooms</a>
          </b-card-footer>
        </b-card>
      </div>
      <!-- Page navigation -->
      <div
        class="overflow-auto"
        v-if="!loading"
        style="justify-content:center; display:flex;padding-bottom:40px;"
      >
        <div>
          <b-pagination-nav
            v-model="currentPage"
            :number-of-pages="pages"
            base-url="#"
            first-number
            size="lg"
            class="nav-style"
          ></b-pagination-nav>
        </div>
      </div>
    </div>
  </div>
</template>
<script >
import Vue from "vue";
import axios from "axios";
import Loader from "../Loader";
import NavBar from "../NavBar";
// import Footer from "../Footer";

export default Vue.extend({
  components: {
    Loader,
    NavBar
    //Footer
  },
  data() {
```

```
  return {
    hotels: [],
    loading: true,
    nutritions: [],
    pages: 4,
    currentPage: 1,
    detailsShow: false,
    hasRoomCleaning: undefined,
    hasParking: undefined,
    stars3: null,
    stars4: null,
    stars5: null,
    distance1: null,
    distance2: null,
    distance3: null,
    ro: null,
    bb: null,
    hb: null,
    fb: null,
    ai: null
  };
},
mounted() {
  axios
    .get("https://localhost:5001/api/hotels/")
    .then(response => {
      setTimeout(() => {
        this.hotels = response.data;
        this.hotels.sort(function(a, b) {
          return b.id - a.id;
        });
        this.loading = false;
      }, 600);
    })
    .catch(error => {
      console.log(error);
    });
  axios
    .get("https://localhost:5001/api/nutritionTypes/")
    .then(response => {
      this.nutritions = response.data;
    })
    .catch(error => {
      console.log(error);
    });
},
methods: {
  getNutritionByHotelId(nutritionTypeId) {
    return this.nutritions[nutritionTypeId - 1].name;
  },
  // goToAdd() {
```

```
//   this.$router.push("/hotels/add");
// },
// gotToDelete() {
//   this.$router.push("/delete");
// },
// goToUpdate() {
//   this.$router.push("/update");
// },
goToRooms(hotelId) {
  return `hotels/${hotelId}/rooms`;
},
ratingColor(rating) {
  if (rating < 3) {
    return "color:red;";
  } else if (rating >= 3 && rating < 5) {
    return "color:#F5C60E;";
  }

  return "color:#00A51B;";
},
setHotelId(hotelId) {
  localStorage.setItem("hotelId", hotelId);
},
filter() {
  let hasParking = this.hasParking;
  let hasRoomCleaning = this.hasRoomCleaning;
  let hotelsTmp = this.hotels;
  if (this.stars3 != 0 || this.stars4 != 0 || this.stars5 != 0) {
    axios
      .get(
        "https://localhost:5001/api/hotels/ratingArray/" +
          this.stars3 +
          "/" +
          this.stars4 +
          "/" +
          this.stars5
      )
      .then(response => {
        setTimeout(() => {
          this.hotels = response.data;
          this.loading = false;
        }, 600);
        hotelsTmp = response.data;
      })
      .catch(error => {
        console.log(error);
      });
  }
  this.hotels = this.hotels.filter(function(x) {
    if (hotelsTmp.indexOf(x) != -1) return true;
    return false;
```

```
});
hotelsTmp = this.hotels;
if (
  this.ro != null ||
  this.bb != null ||
  this.hb != null ||
  this.fb != null ||
  this.ai != null
) {
  axios
    .get(
      "https://localhost:5001/api/hotels/nutritionArray/" +
        this.ro +
        "/" +
        this.bb +
        "/" +
        this.hb +
        "/" +
        this.fb +
        "/" +
        this.ai
    )
    .then(response => {
      setTimeout(() => {
        this.hotels = response.data;
        this.loading = false;
      }, 600);
    })
    .catch(error => {
      console.log(error);
    });
}
this.hotels = this.hotels.filter(function(x) {
  if (hotelsTmp.indexOf(x) != -1) return true;
  return false;
});
hotelsTmp = this.hotels;
if (hasRoomCleaning != undefined) {
  axios
    .get(
      "https://localhost:5001/api/hotels/roomCleaning/" + hasRoomCleaning
    )
    .then(response => {
      setTimeout(() => {
        this.hotels = response.data;
        this.loading = false;
      }, 600);
    })
    .catch(error => {
      console.log(error);
    });
```

```
}
this.hotels = this.hotels.filter(function(x) {
  if (hotelsTmp.indexOf(x) != -1) return true;
  return false;
});
hotelsTmp = this.hotels;
if (hasParking != undefined) {
  axios
    .get("https://localhost:5001/api/hotels/parking/" + hasParking)
    .then(response => {
      setTimeout(() => {
        this.hotels = response.data;
        this.loading = false;
      }, 600);
    })
    .catch(error => {
      console.log(error);
    });
}
this.hotels = this.hotels.filter(function(x) {
  if (hotelsTmp.indexOf(x) != -1) return true;
  return false;
});
hotelsTmp = this.hotels;
if (
  this.distance1 != null ||
  this.distance2 != null ||
  this.distance3 != null
) {
  axios
    .get(
      "https://localhost:5001/api/hotels/distance/" +
        this.distance1 +
        "/" +
        this.distance2 +
        "/" +
        this.distance3
    )
    .then(response => {
      setTimeout(() => {
        this.hotels = response.data;
        console.log(this.hotels);
        this.loading = false;
      }, 600);
    })
    .catch(error => {
      console.log(error);
    });
}
this.hotels = this.hotels.filter(function(x) {
  if (hotelsTmp.indexOf(x) != -1) return true;
```

```
        return false;
      });
    },
    resetAll() {
      var radio1 = document.getElementsByName("radio1");
      var radio2 = document.getElementsByName("radio2");
      var radio3 = document.getElementsByName("radio3");
      var radio4 = document.getElementsByName("radio4");
      radio1[0].checked = false;
      radio2[0].checked = false;
      radio3[0].checked = false;
      radio4[0].checked = false;
      var nutritionCBs = document.getElementsByName("nutrition");
      for (let i = 0; i < nutritionCBs.length; i++) {
        nutritionCBs[i].checked = false;
      }
      var distanceCBs = document.getElementsByName("distance");
      for (let i = 0; i < distanceCBs.length; i++) {
        distanceCBs[i].checked = false;
      }
      var starsCBs = document.getElementsByName("stars");
      for (let i = 0; i < starsCBs.length; i++) {
        starsCBs[i].checked = false;
      }
      this.stars3 = undefined;
      this.stars4 = undefined;
      this.stars5 = undefined;
      this.hasParking = undefined;
      this.hasRoomCleaning = undefined;
      this.distance1 = null;
      this.distance2 = null;
      this.distance3 = null;
      axios
        .get("https://localhost:5001/api/hotels/")
        .then(response => {
          setTimeout(() => {
            this.hotels = response.data;
            this.hotels.sort(function(a, b) {
              return b.id - a.id;
            });
            this.loading = false;
          }, 600);
        })
        .catch(error => {
          console.log(error);
        });
    }
  }
});
</script>
```

```
<style>
.hotelCard {
  font-weight: bold;
  font-size: 16px;
  font-family: Cambria, Cochin, Georgia, Times, "Times New Roman", serif;
  justify-content: center;
  align-items: center;
  margin-bottom: 25px;
  display: flex;
}
span {
  padding-left: 10px;
}
ul {
  list-style: none;
  margin: 0;
  padding: 0;
}
.rating-style {
  font-weight: bold;
  font-size: 22px;
}
.nav-style {
  border: 1px solid lightgray;
  background-color: white;
}
.nav-style li {
  border: 1px solid lightgray;
}
.title-design {
  text-align: center;
  font-family: Cambria, Cochin, Georgia, Times, "Times New Roman", serif;
  margin: 50px auto 20px auto;
  font-weight: bold;
}
.filer {
  width: 250px;
  height: auto;
}
.img-fluid {
  max-width: 100%;
  height: 255px;
}
.more-info {
  display: flex;
  justify-content: space-between;
  font-size: x-large;
  cursor: pointer;
}
</style>
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів

Програма та методика тестування

КПІ.ІП-6311.045440.04.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ К.І. Ліщук

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Р. Л. Заєць

Київ – 2020 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Веб-застосування для бронювання готелів, яке являє собою веб-сайт, створенний на платформі .NET з використанням технології ASP .NET Web Api Core та Vue.js.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

– функціональна працездатність елементів сторінок веб-ресурсу;

– можливість здійснення бронювання користувачем;

– можливість ведення готелів, кімнат на фільтрів адміністратором;

– можливість фільтрації готелів за вибраними критеріями;

– можливість перегляду та виставлення рейтингу користувачу менеджером готелю;

– забезпечення належного рівня безпеки даних;

– зручність роботи з веб-сайтом;

– відповідність дизайну вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

– функціональне тестування, зокрема на рівні Critical path test (базове тестування);

– тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);

– тестування інтерфейсу.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність веб-ресурсу перевіряється шляхом:

– динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування веб-ресурсу в різних веб-браузерах;

– тестування при максимальному навантаженні;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів

Керівництво користувача

КПІ.ІП-6311.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ К.І. Ліщук

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ Р.Л. Заєць

Київ – 2020 року

ЗМІСТ

1 КОРИСТУВАЧ

Щоб мати змогу використовувати веб-застосування, користувач повинен відкрити його в будь-якому браузері серед наступних:

- Mozilla Firefox;
- Google Chrome;
- Safari;
- Internet Explorer;
- Opera Browser.

Вхід в систему.

Користувач починає свою роботу з сторінки авторизації. Йому потрібно ввести правильні електронну адресу та пароль і натиснути кнопку «*LOGIN*». Щоб система запам'ятала даного користувача, необхідно перед натисненням кнопки «*LOGIN*» натиснути на чекбокс «*Remember me*»

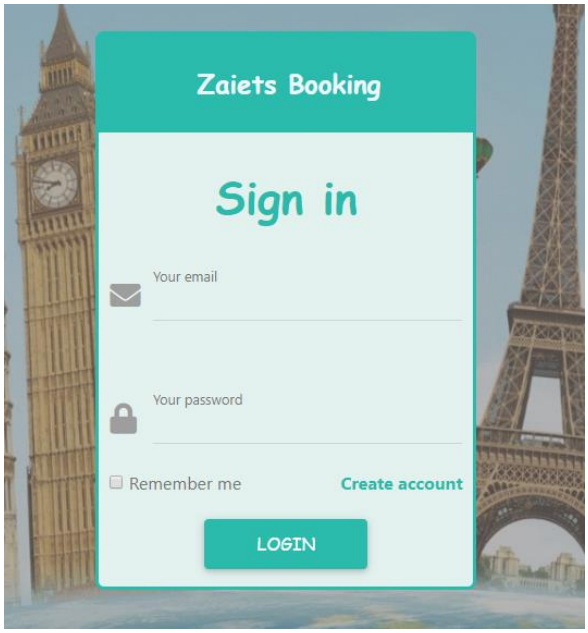


Рисунок 1 – Панель авторизації користувача

Якщо ще не було зареєстровано власного акаунту, то потрібно натиснути кнопку «*Create account*». У користувача відкриється нове вікно з формою для

реєстрації. Користувач попинен ввести дані у відповідні поля та натиснути кнопку «*CREATE*». Якщо введені дані пройшли валідацію, система виведе на екран відповідне повідовлення про успіх або невдачу. Після успішного ствоення акаунту користувач повинен натиснути кнопку «*SIGN IN*» та авторизуватися.

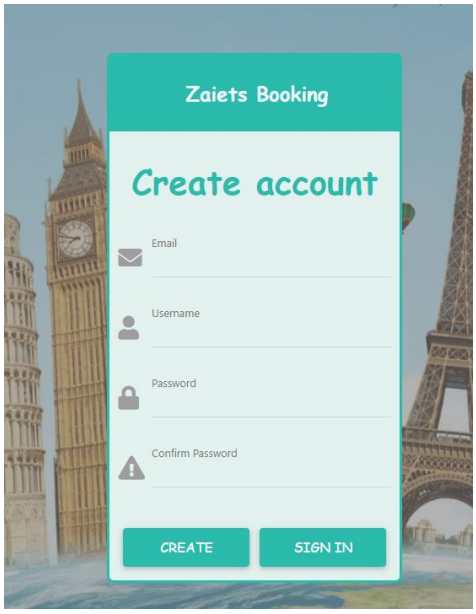


Рисунок 2 – Панель реєстрації користувача

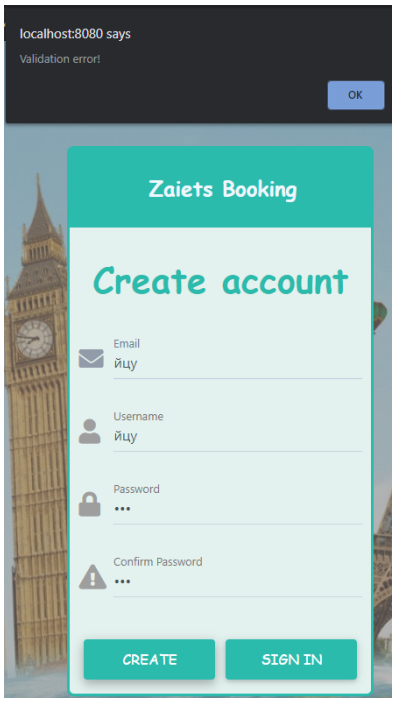


Рисунок 3 – Повідомлення системи про невдале створення користувача

Рисунок 4 – Повідомлення системи про успішне створення користувача

Головна сторінка.

Рисунок 5 – Головна сторінка веб-застосування

Потрапивши на головну сторінку, користувач має змогу до перети до перегляду готелів, натиснувши кнопку «*Hotels*», до сторінки власного профілю, натиснувши кнопку «*ACCOUNT*» або вийти з системи – «*LOG OUT*». Дані кнопки доступну на панелі навігації.

Щоб розпочати бронювання, користувач повинен вибрати дати для подорожі та натиснути кнопку «*TO HOTELS*»

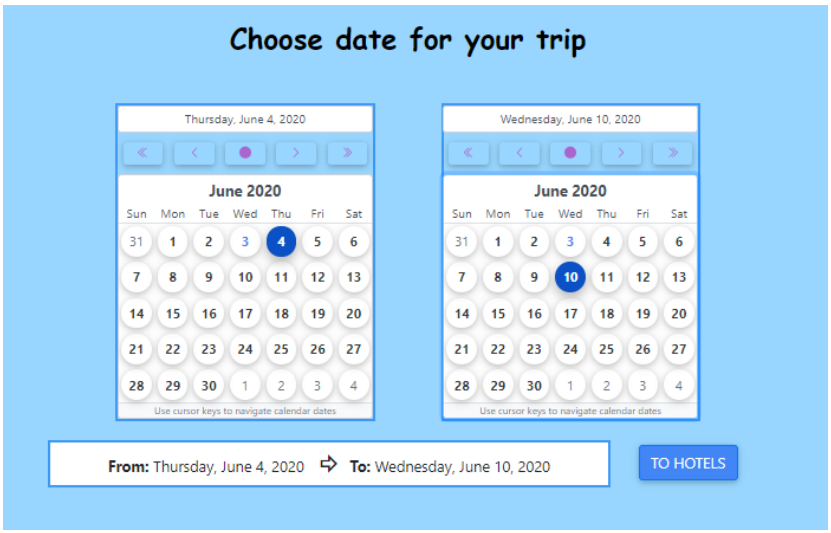


Рисунок 6 – Вибір дати заїзду та від'їзду з готелю майбутньої подорожі

Після натиснення на кнопку «*TO HOTELS*» , користувач потрапляє на сторінку з готелями. За допомогою зручного фільтру, користувач має змогу отримати саме ті готелі, які задовольняють його потреби.

Сторінка готелів.

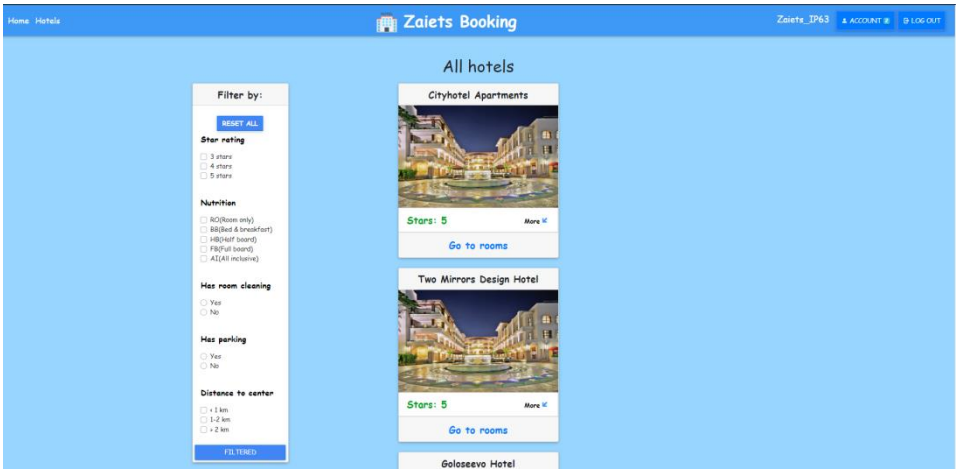


Рисунок 7 – Строрінка з готелями та фільтрами

Щоб розгорнути картку готелю та побачити детальну інформацію, користувач повинен натиснути на стрілку, поряд с написом «*More*» More

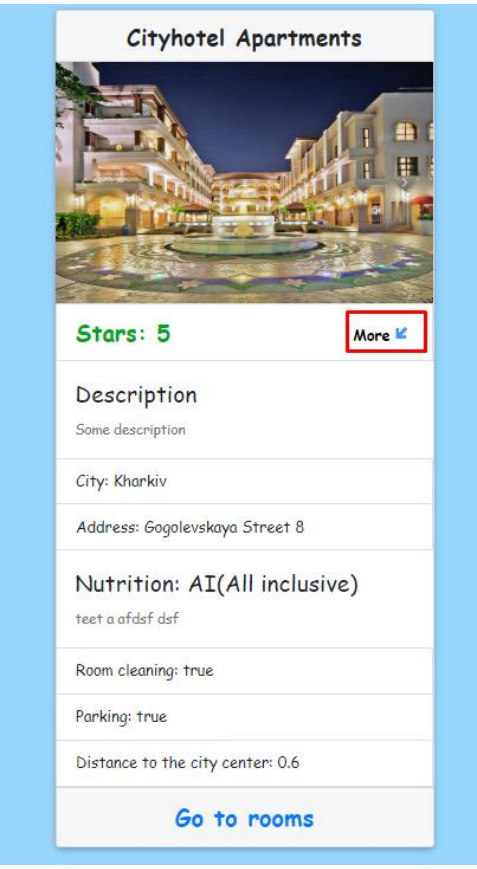


Рисунок 8 – Результат натиснення на стрілку

Фільтр має дві кнопку «*RESET ALL*» - для очищення форми фільтрації та «*FILTERED*» - для виконання фільтрації. Також користувач має змогу відфільтрувати готелі за наступними критеріями : за кількістю зірок , за типом харчування, чи присутнє прибирання кімнати, чи присутня парковка для автомобілів, за відстанню від готелю до центру міста.

Щоб відфільтрувати готелі, користувач повинен натиснути на певні критерії, а потім на кнопку «*FILTERED*»

Рисунок 9 – Фільтр

Користувач має змогу перейти до номерів готелю натиснувши кнопку «*Go to rooms*»

Для навігації між сторінками, користувач має змогу користуватися навігатором сторінок.

Рисунок 10 – Навігатор сторінок

Сторінка з номерами.

Рисунок 11 – Сторінка номерів

На сторінці з номерами, перед користувачем постають доступні та недоступні кімнати. Усі номери сірого кольору є недоступними до бронювання.

Якщо користувач визначимся с з типом кімнати, він може забронювати її натискаючи на кнопку «*RESERVE*».

Форма бронювання номеру.

Рисунок 12 – Форма бронювання

На формі бронювання користувач повинен внести правильні дані у відповідні поля та натиснути кнопку «*RESERVE*»

Сторінка власного акаунту.

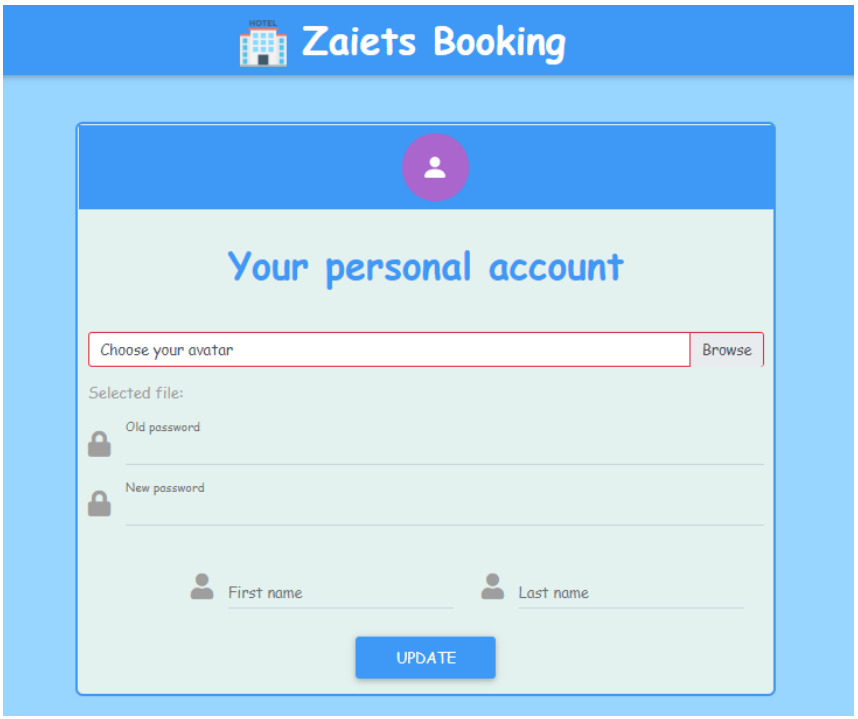


Рисунок 13 – Сторінка акаунту

На сторінці акаунту користувач має змогу змінити свій пароль, загрузити собі фотографію до профілю та встановити записати своє прізвище та ім'я.

2 АДМІНІСТРАТОР

Адміністратору доступні сторінки з додавання, оновленням та видаленням номерів та готелів.

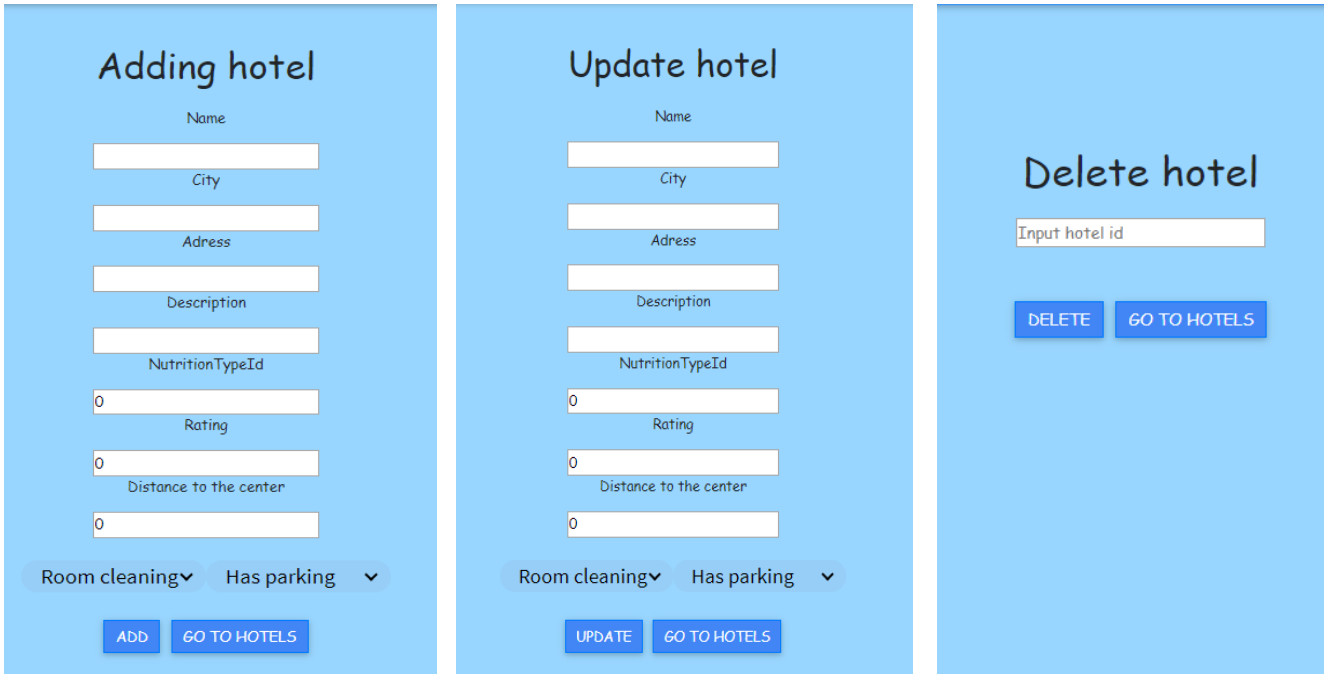


Рисунок 14 – Сторінки ведення готелів

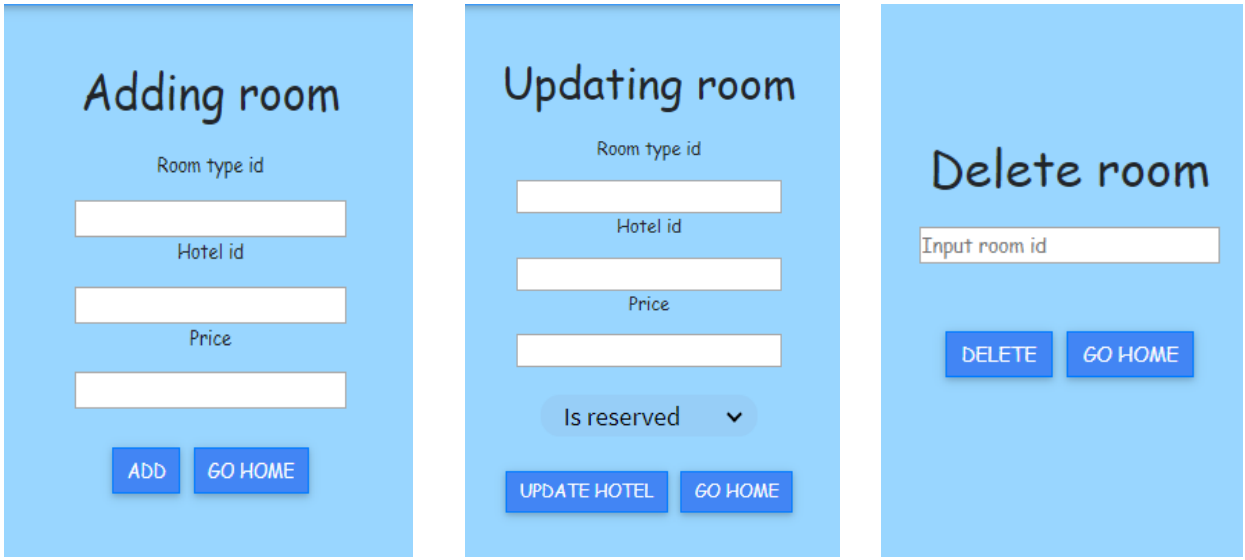


Рисунок 15 – Сторінки ведення номерів

На сторінках ведення готелів, адміністратор після введення необхідних даних повинен натиснути відповіднку кнопку «*ADD*», «UPDATE», «DELETE». У разі успіху, для повернення до готелів, повинен натиснути кнопку «*GO TO HOTELS*».

На сторінках ведення номерів, адміністратор після введення необхідних даних повинен натиснути відповіднку кнопку «*ADD*», «UPDATE», «DELETE». У разі успіху, для повернення до головної, повинен натиснути кнопку «*GO HOME*».

3 МЕНЕДЖЕР ГОТЕЛЮ

Менеджер готелю має змогу дивитися оцінки та відгуки про клієнта та підтвердити або скасувати бронювання. Кнопка «ACCEPT» - підтвердити та «DISCARD» - скасувати.

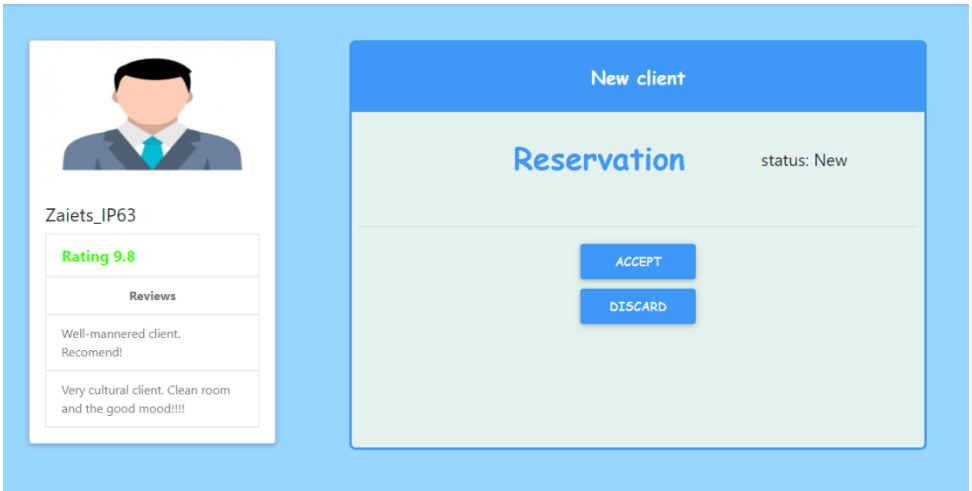


Рисунок 16 – Сторінка з новим бронюванням для менеджера готелю

Після перебування клієнта в готелі, менеджер має змогу оцінити його та залишити коментар, який побачать менеджери інших готелів.

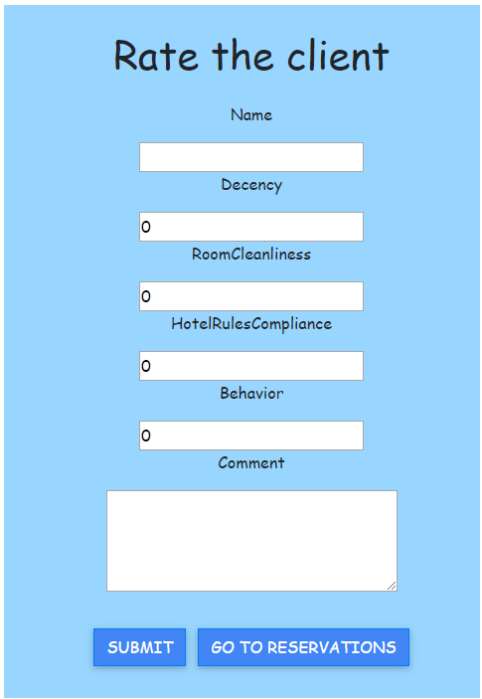
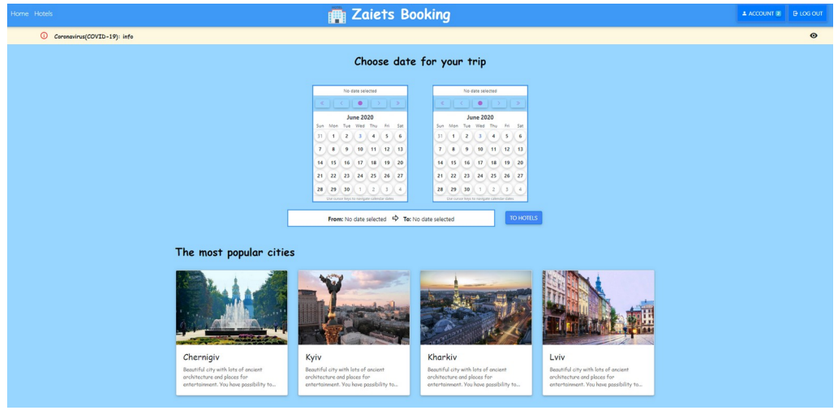
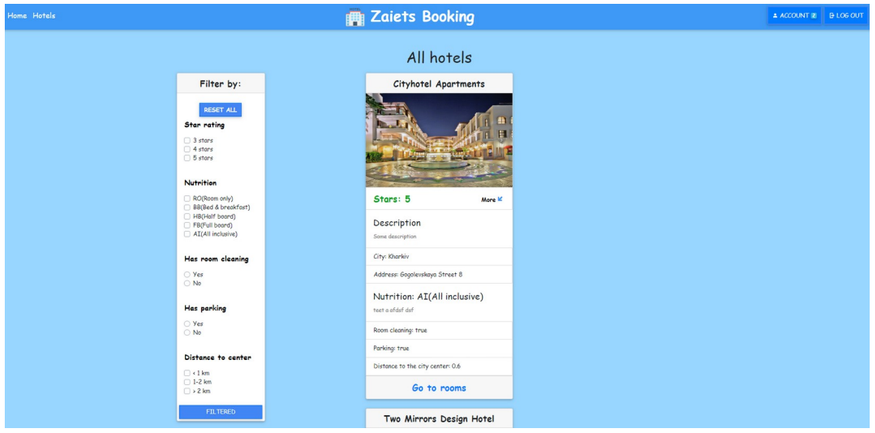
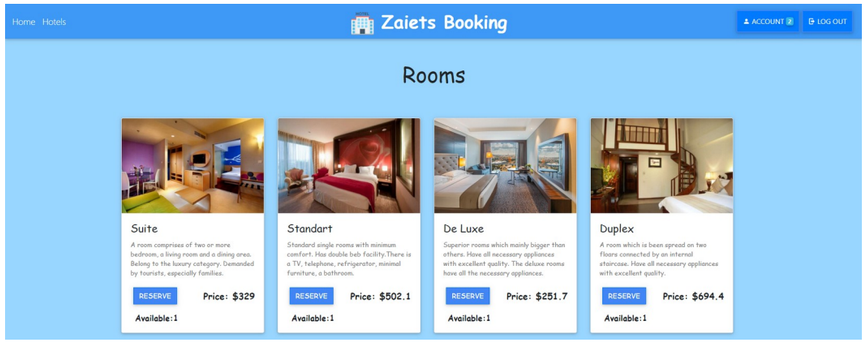


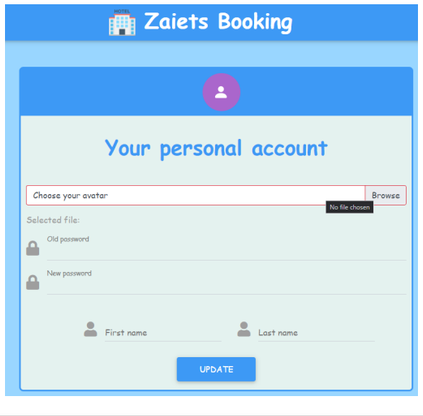
Рисунок 17 – Сторінка оцінювання клієнта

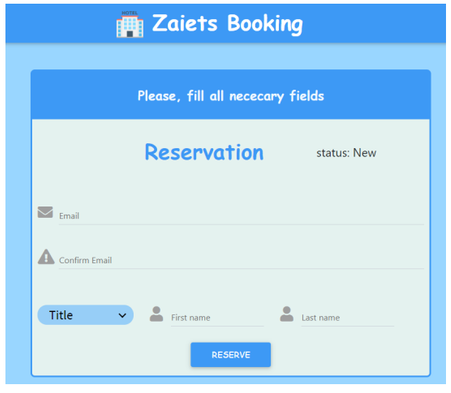
КПІ.ІП-6311.045440.06.99.КЕ

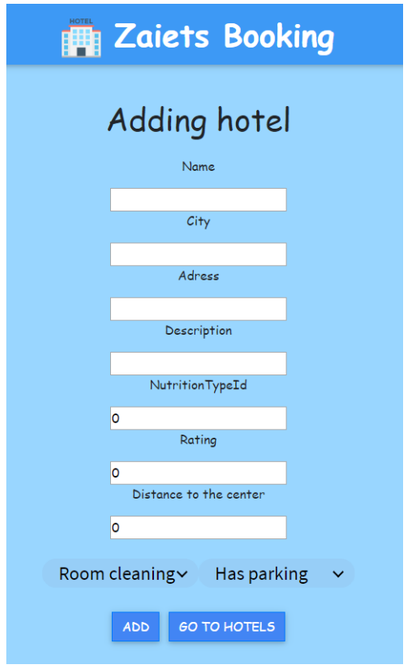


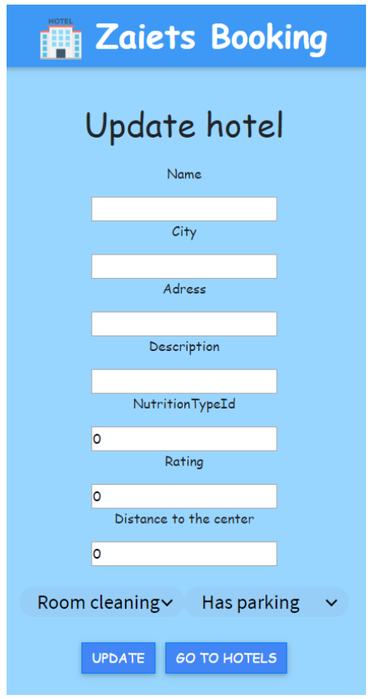


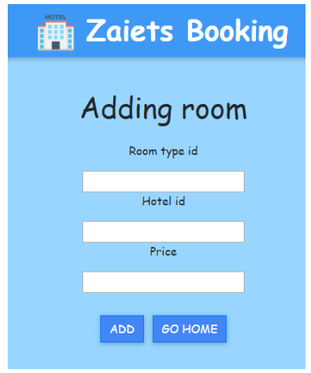


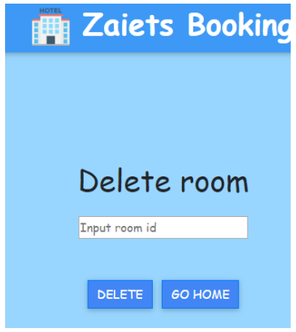


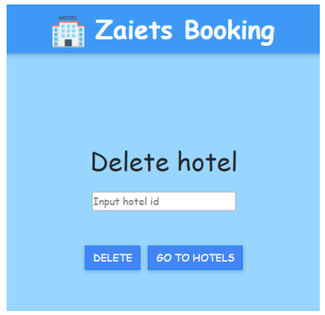


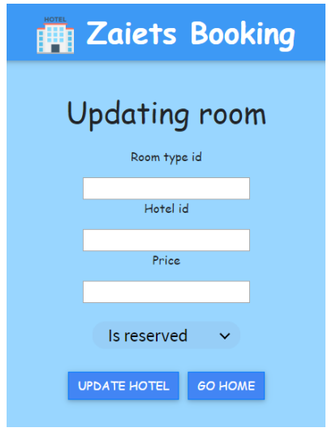


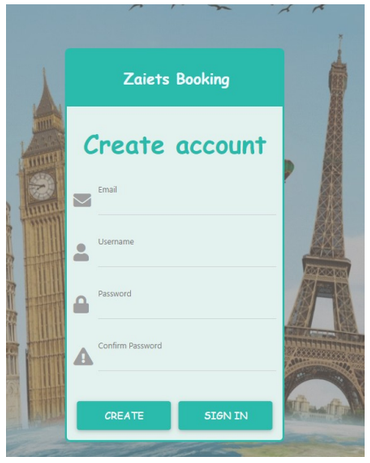


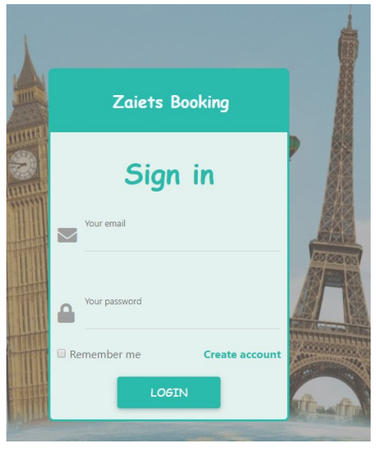


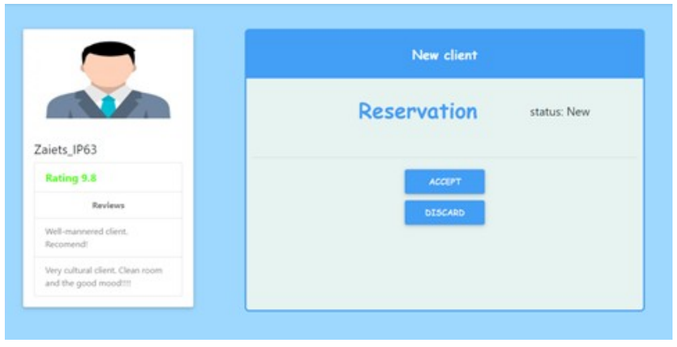


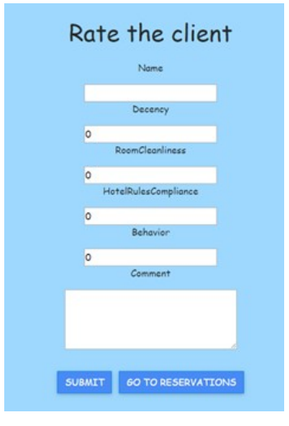




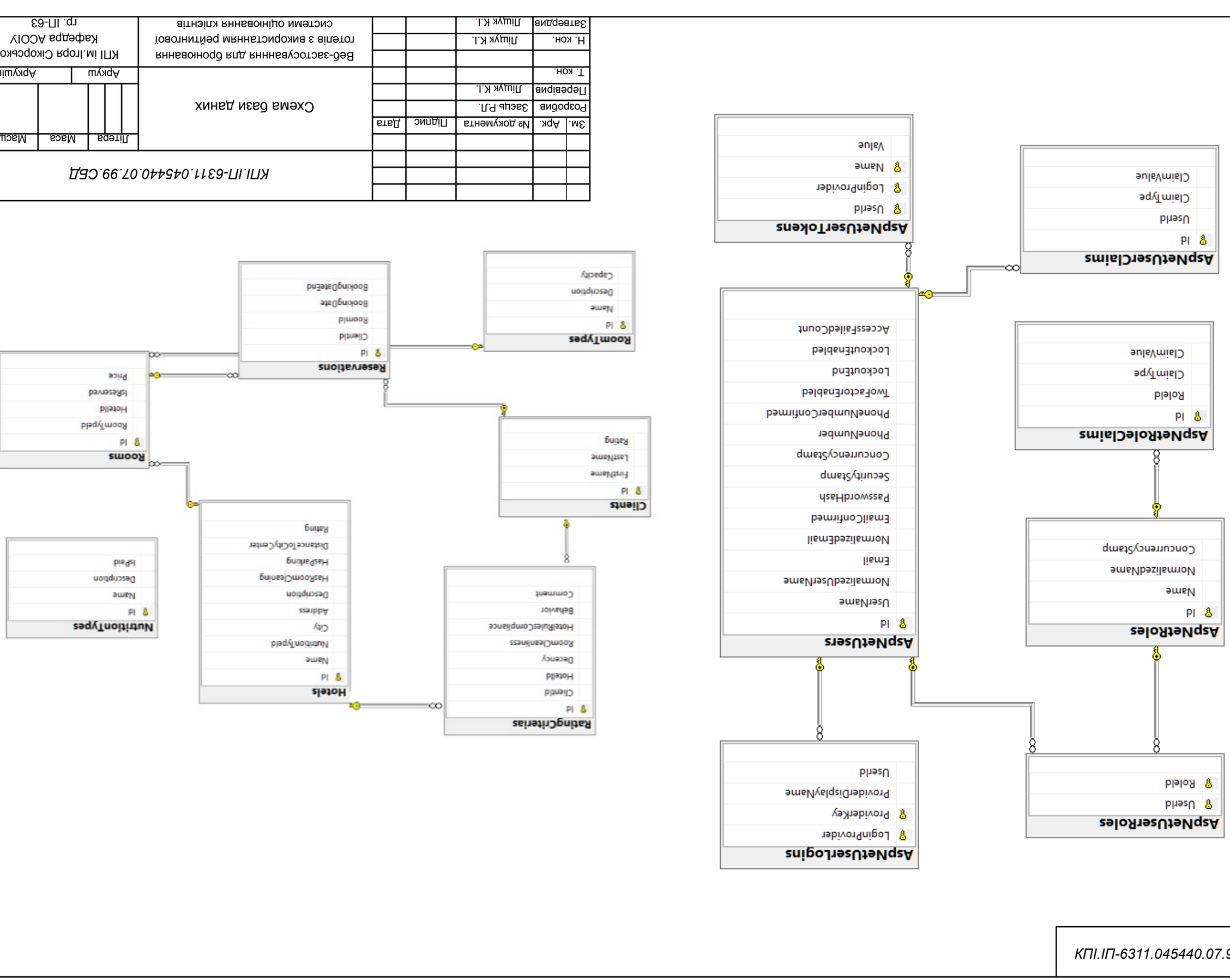








					КПІ.ІП-6311.045440.06.99.КЕ		
					Креслення вигляду екранних форм	Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Заєць Р.Л.				Аркуш	Аркушів
Перевірив		Ліщук К.І.					
Т. кон.							
Н. кон.		Ліщук К.І.			Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63	
Затвердив		Ліщук К.І.					



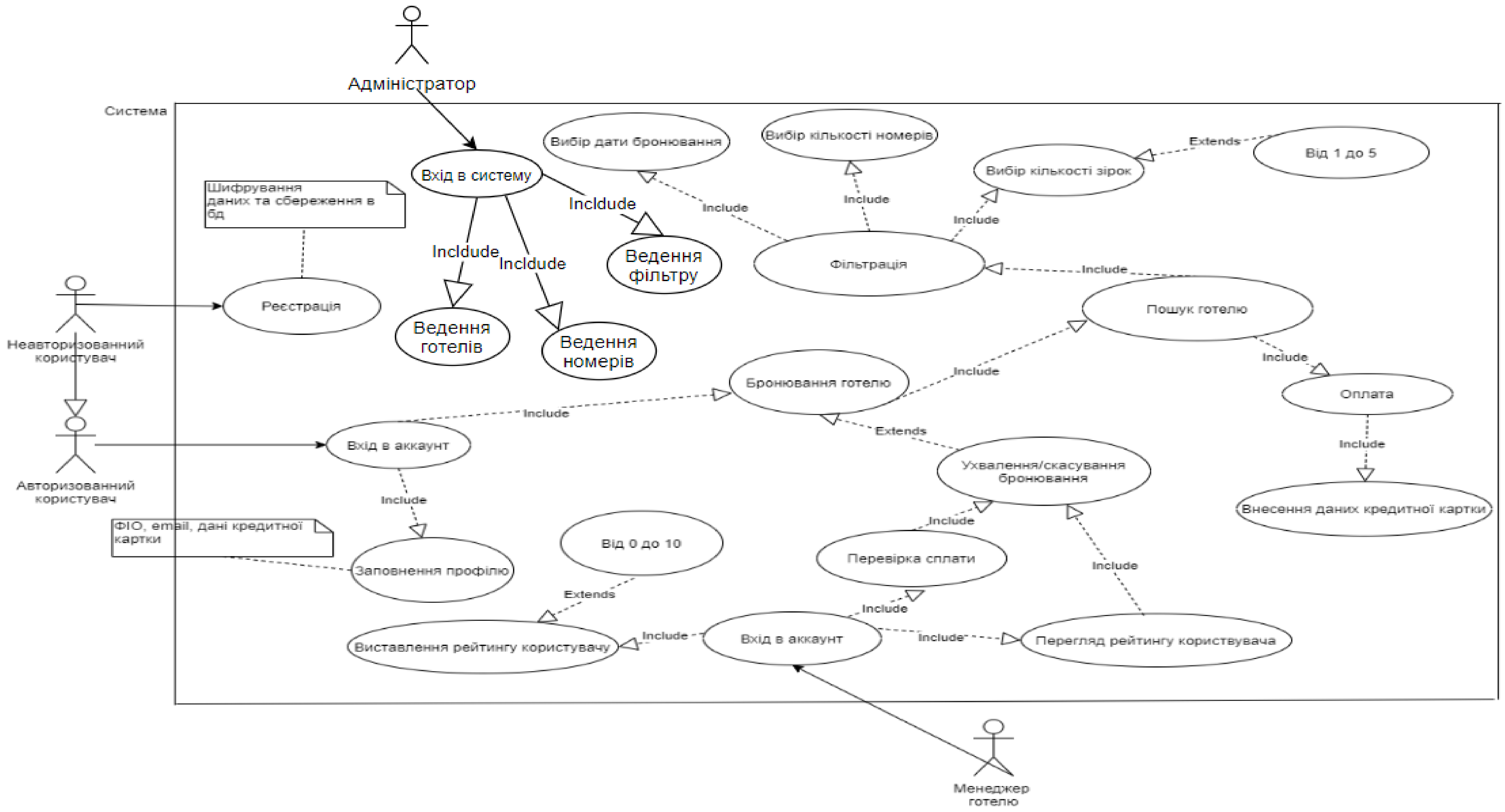
Зм.	Арк.	№ докумeнта	Підпис	Дата	
Розробив		Засць Р.Л.			Схема бази даних
Перевірив		Лішук К.І.			
Т. кон.					
Н. кон.		Лішук К.І.			Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів
Затвердив		Лішук К.І.			

Літера	Маса	Масштаб
Аркуш		Аркушів

КПІ ім.Ігоря Сікорського
Кафедра АСОІУ
гр. ІП-63

КПІ.ІП-6311.045440.07.99.СБД

КПІ.ІП-6311.045440.08.99.СС



					КПІ.ІП-6311.045440.08.99.СС
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання
Розробив		Заєць Р.Л.			Літера / Маса / Масштаб
Перевірив		Ліщук К.І.			
Т. кон.					Аркуш / Аркушів
					Веб-застосування для бронювання готелів з використанням рейтингової системи оцінювання клієнтів
Н. кон.		Ліщук К.І.			КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63
Затвердив		Ліщук К.І.			